Capítulo 1 – Conceitos Básicos de Análise de Sistemas e Projetos

Este capítulo apresenta uma visão geral sobre a evolução dos sistemas e da Engenharia de Software, iniciando com a narrativa da crise do software e como essa crise foi resolvida, ou pelo menos minimizada com a adoção de metodologias. Aborda também questões relacionadas à modelagem e sua importância para o desenvolvimento de sistemas.

1.1. A Crise do Software

No Início do desenvolvimento de software, mais especificamente a partir dos anos 70, existia uma forte demanda por sistemas, no entanto, as técnicas existentes para sua produção, aliadas a baixa produtividade e complexidade pertinentes ao processo, gerou muitos problemas de qualidade e manutenção dos sistemas.

O fato de desenvolver sistemas de baixa qualidade que, muitas vezes, não atendia as necessidades dos usuários, gerou o que passou a ser chamado de Crise do Software. Nessa época os sistemas eram produzidos sem muito controle, dependentes totalmente da competência dos profissionais. Obviamente isso fez aumentar a demanda por esses profissionais. Mas, isso não era suficiente.

Como os sistemas foram crescendo e se tornando cada vez mais complexos, houve a necessidade de aumentar o controle sobre as tarefas pertinentes à produção dos sistemas. Se fizermos uma analogia de um sistema produzido, no final da década de 80 com os dias atuais, veremos que a complexidade vem aumentando ano a ano. Esse foi o motivo para o surgimento da Engenharia de Software, uma tentativa de controlar a complexidade envolvida no desenvolvimento de um sistema.

Podemos dizer que os elementos responsáveis pela redução dos problemas trazidos pela crise do software foram os seguintes: a implementação de processos sistematizados, uso de técnicas e ferramentas apropriadas, treinamento contínuo da equipe e mudança na postura dos profissionais, reconhecendo que a presença do cliente durante o processo de desenvolvimento é essencial para o sucesso da grande maioria dos projetos.

Ainda assim, é importante afirmar que os problemas na produção de software continuam os mesmos para os dias atuais, boa parte dos projetos continuam gerando problemas semelhantes. Mais do que nunca, a utilização de ferramentas e processos

padronizados deve ser implementados continuamente, em especial nas empresas produtoras de sistemas.

1.2. Surgimento da Engenharia de Software

Como citamos na seção anterior, a Engenharia de Software teve como objetivo principal reduzir os impactos negativos provenientes da crise do software. Inspirada nas engenharias existentes, a Engenharia de Software define diversas metodologias, focadas no planejamento, implementação e gerenciamento do processo pertinentes ao desenvolvimento de sistemas.

A Engenharia de Software alterou a forma de trabalho, até então artesanal, usada pelos profissionais no desenvolvimento de sistemas, buscando tornar o trabalho mais contínuo, sequencial, que segue um conjunto de passos bem definidos. Isso reduz o lado pessoal e alinha a equipe para torná-la mais eficiente, eficaz e, principalmente, menos dependente de um único membro da equipe.

Atualmente, um profissional que trabalha com Engenharia de Software é alguém capaz de planejar, projetar, modelar, fazer testes e manutenções em diversos tipos de sistemas, sejam do tipo desktop, aplicativos ou para a Internet, não importa o tipo do sistema, os procedimentos e técnicas a serem utilizados são praticamente os mesmos.

Dessa forma, é de se esperar que um Engenheiro de Software esteja apto para atuar nas mais diferentes etapas de um processo de desenvolvimento, como veremos daqui por diante. As atividades mais comuns associadas a esse profissional se concentram no levantamento e especificação de requisitos, no gerenciamento do projeto, definição da arquitetura do sistema, elaboração da documentação entre outras atividades.

1.3. Problemas no desenvolvimento de sistemas

A produção de um sistema, apesar de conter passos e processos semelhantes a criação de produtos físicos, é bastante diferente e desafiadora. Mesmo softwares considerados de sucesso apresentam problemas e bugs de maneira frequente, mostrando que a complexidade, inerente ao processo de desenvolvimento e manutenção, se mostra cada vez mais presente.

Se você pesquisar na Internet pelos termos “software is hard” encontrará diversos artigos e blogs refletindo a respeito das particularidades e dos desafios existentes na produção do software. Uma pesquisa do Chaos Report (1994), alertava para diversos problemas relacionados ao desenvolvimento de aplicações, dos quais destacamos apenas cinco itens.

1. Apenas 10% dos projetos terminam dentro do prazo estimado.
2. Aproximadamente 25% dos projetos são descontinuados.
3. Mais de 50% dos projetos possuem um custo acima do esperado.
4. Apenas 2% das aplicações podem ser usadas quando entregue.
5. Quase 50% do software entregue nunca foi usado.

Apesar de o relatório ser antigo, quase 30 anos atrás, o desenvolvimento atual de sistemas ainda pode passar por problemas semelhantes.

A questão do prazo citada no item 1 continua sendo verdade para uma boa parte dos projetos, uma vez que mudanças na equipe, no orçamento ou mesmo nos requisitos, fazem com que o sistema não seja entregue no prazo estimado.

Em relação ao item 2, ainda existem muitos projetos com este problema por diversos fatores: saída de membros chave da equipe, falta de experiência com a tecnologia envolvida, mudanças políticas, etc.. Parar um projeto antes de terminar pode ser bastante frustrante para a equipe, pois quem trabalha com desenvolvimento sabe o quanto é importante ver seu software rodando. Dedicar um ano na produção de algo que sequer será utilizado, pode causar muita decepção entre os profissionais.

Já o item 3 destaca a questão do custo, fator surpresa que pode mudar com o passar do tempo, em especial se o projeto for grande (e a equipe pequena). Isso mostrava na época e, pode ser aplicado ainda hoje, que a falta de compreensão do todo em relação ao que deve ser produzido e entregue pelo sistema, pode gerar trabalho adicional e, consequentemente, alterar o custo. Normalmente, o aumento do custo está associado a mudanças de requisitos. Conforme o sistema vai sendo desenvolvido, novas necessidades aparecem.

O conteúdo do item 4 era uma verdade absoluta para a época, principalmente em função da metodologia de desenvolvimento usada no início da engenharia do software: a metodologia em cascata. Vamos tentar resumir essa forma de produção

de software. Acreditava-se que para se produzir um sistema era necessário apenas seguir um conjunto de passos sequenciais, semelhante a fazer um produto físico qualquer. Como veremos, as fases para desenvolvimento de um sistema existem até hoje, o que mudou foi a maneira de conduzir esse processo. Por diversas razões, antigamente, um analista de sistemas visitava (fisicamente) a empresa cliente interessada no sistema (que poderia estar localizada em outro estado), coletava todas as informações relacionadas ao sistema, retornava a seu escritório e iniciava a produção do sistema. Depois de um tempo considerável (meses ou mesmo anos), retornava “com o sistema embaixo do braço” e implantava para ser usado. Atualmente é fácil de entender que essa forma de trabalho não tem como dar certo! Vamos descrever melhor esse processo na seção 4 do capítulo 1, relacionado ao manifesto ágil.

No item 5 é abordada uma questão que, provavelmente, é muito semelhante ao cenário atual - praticamente metade dos sistemas produzidos não era utilizada. Imagine se isso ocorresse com automóveis, aviões, pontes, edifícios etc. Quanto prejuízo! Muitas vezes um sistema não é utilizado por mínimos detalhes. Talvez a falta de precisão em cálculos, a falta de comunicação com algum outro sistema, ou ainda, simplesmente porque o usuário não gostou “da cara” da interface gráfica.

Se considerarmos que a maioria dos aplicativos desenvolvidos para smartphones não obtém sucesso atualmente, então talvez nossa estatística tenha piorado ainda mais! Milhares de projetos e sistemas são publicados diariamente nas “stores”, apenas alguns fazem sucesso. Como dissemos, produzir um software é uma tarefa difícil, fazer com que um software se torne um sucesso é mais difícil ainda. Mas, tudo bem, para nós da área da computação, o que nos move são os desafios!

1.4. Manifesto Ágil

Elaborado a partir do ano 2000, o manifesto ágil se refere à uma “declaração de valores e princípios essenciais para o desenvolvimento de software”. Foi um tipo de reunião ocorrida em 2001, em que participantes experientes da área de desenvolvimento de sistemas, apresentaram os aspectos positivos de algumas metodologias que vinham utilizando. O evento gerou um momento de mudança de paradigma, isto é, uma mudança na maneira de conduzir o processo de desenvolvimento.

O manifesto ágil definiu quatro postulados descritos nos parágrafos seguintes.

Postulado 1 - Indivíduos e interações mais do que processos e ferramentas:

Os processos de desenvolvimento de software existentes definiam uma disciplina rígida que deveria ser seguida à risca em todas as fases. Segundo os participantes do manifesto ágil, essas regras nem sempre contribuem para aumentar a qualidade e velocidade em relação a produção do software. Desta forma, foi considerado na época que as interações entre os participantes da equipe eram mais eficientes do que seguir um plano rígido definido pelo processo de desenvolvimento. Por causa disso, uma das características presentes nas metodologias de desenvolvimento ágeis é a presença da reunião diária - uma maneira de manter a equipe focada e atualizada quanto às alterações exigidas dentro do processo.

Postulado 2 - Trabalhar no software mais do que documentação abrangente:

A maioria dos profissionais de desenvolvimento de software (os programadores) gostam mais de trabalhar na codificação de um sistema ao invés de realizar sua documentação. Esse segundo postulado descreve que é mais importante trabalhar no software do que manter constantemente atualizada a documentação do sistema. Nas metodologias da época era comum serem exigidas muitas documentações, pois se acreditava que ao documentar existiam maiores possibilidades de organizar o desenvolvimento e, consequentemente, aumentar a qualidade do sistema e também as chances de sucesso. No entanto, com o desenvolvimento de novas ferramentas que facilitam o compartilhamento e a integração, tanto do desenvolvimento de um sistema, quanto de sua documentação, manter a equipe atualizada tornou-se muito mais simples. É importante dizer que o manifesto ágil não descarta a documentação, é uma questão de prioridade, melhor trabalhar no software do que investir um tempo exagerado em sua documentação.

Postulado 3 - Colaboração do cliente mais do que negociação contratual:

Esse postulado considera essencial a participação do cliente durante todas as etapas da produção do sistema. Manter o cliente junto à equipe é um procedimento que pode representar uma economia considerável, de tempo e recursos, no momento de produzir um novo software. Isso é verdade na maioria dos casos porque

os desenvolvedores nem sempre conhecem o processo no qual o sistema será inserido. Por outro lado, espera-se que o cliente tenha toda a experiência necessária (no processo) para orientar a equipe de desenvolvimento. Produzir um sistema sem a presença constante do cliente é “viver na corda bamba”, é correr o risco de ter retrabalho e, consequentemente, aumento nos custos de produção. No século passado, muitas vezes o cliente era visto como um vilão, pois vivia exigindo alterações depois que o sistema estava pronto! Felizmente isso mudou, hoje o cliente é visto como um parceiro.

Postulado 4 - Responder às mudanças mais do que seguir um plano:

Essa premissa talvez resuma a proposta do manifesto ágil. Quando falamos a respeito de metodologia ágil, o termo ágil não representa, necessariamente, o fato de produzir um software rapidamente. Esse termo está mais ligado com a velocidade com o que a equipe e, consequentemente, o sistema, deve responder às mudanças durante as etapas do desenvolvimento. Nas metodologias anteriores, o plano a ser seguido era definido no início, desde a concepção do sistema, e deveria ser seguido rigidamente. Atualmente, num momento em que as mudanças são contínuas, seguir um plano desde o início, sem poder fazer ajustes, parece um contrassenso que normalmente não funciona. O mercado exige que as empresas sejam flexíveis, que seus colaboradores sejam flexíveis e também que seus sistemas sejam flexíveis. Enfim, é importante que a equipe envolvida na produção do sistema tenha uma mente aberta, flexível, sempre pronta para mudanças.

Resumidamente, após o manifesto ágil surgiram diversas metodologias de desenvolvimento de software seguindo esses postulados. Podemos considerar que houve uma mudança de paradigma, uma maneira diferente de pensar ou de conduzir o desenvolvimento de um sistema. Essas novas metodologias passaram a produzir um sistema em etapas, em ciclos, ou ainda em iterações. Isso ficará mais claro quando estudarmos as metodologias XP e Scrum, no próximo capítulo.

1.5. Modelagem de Sistemas

A modelagem é utilizada por diversas áreas do conhecimento, em especial no desenvolvimento de sistemas. A modelagem se refere a elaborar algum tipo de representação, um esboço que possa representar um objeto real ou de software. A

modelagem cria uma representação visual de alguma coisa, normalmente para facilitar a compreensão do objeto a ser elaborado. Quanto mais complexo for o objeto, maior será a indicação da modelagem como ferramenta para facilitar o entendimento do que deverá ser produzido.

Ao construir uma residência, por exemplo, podemos citar diversos tipos de modelo: a planta da casa, um esboço da fachada, o esquema elétrico, o esquema hidráulico, entre outros. Cada modelo representa um aspecto diferente do mesmo objeto sendo modelado. Além disso, quanto maior for a casa, ou o prédio, mais importante será o papel da modelagem.

Usando a modelagem, é possível ter uma boa ideia de como o objeto ficará depois de pronto, no caso a casa. O mesmo acontece com um sistema de software: antes de realizar sua construção, podemos elaborar diversos modelos, cada um representando um aspecto diferente do sistema. Um profissional da área de desenvolvimento, tipicamente um analista de sistemas pode elaborar diagramas (modelos) que representam as diferentes partes de um sistema, contendo seus relacionamentos e integrações. Esses diagramas podem ser encaminhados para a equipe de desenvolvimento, mais especificamente para os programadores. Os programadores podem interpretar os diagramas e realizar sua implementação.

A modelagem é mais utilizada nas fases iniciais do desenvolvimento de um sistema, mas pode ser útil a qualquer momento do ciclo de vida de um software. Um modelo pode ser usado para ajudar a entender o que deve ser feito pelo sistema, como para validar se a funcionalidade planejada está funcionando corretamente. A modelagem não é uma “ciência” exata, isso significa que pessoas diferentes podem fazer modelos diferentes para representar o mesmo sistema. Como as necessidades do sistema mudam no decorrer do tempo (às vezes, de um dia para o outro), é muito comum ajustar o modelo durante o ciclo de vida do projeto. Além dessa volatilidade das necessidades, nem sempre o analista possui a visão completa do que deve ser feito no momento em que ele está modelando o sistema.

Como já citado, o processo de modelagem ajuda na compreensão do que deve ser feito, além disso, ao se elaborar diagramas e outros artefatos, ocorre um processo de documentação do sistema. Apesar de ser uma tarefa difícil de ser

realizada, manter a documentação atualizada é muito importante para a equipe de desenvolvimento. Caso um membro da equipe venha a sair e, consequentemente, outro membro comece a fazer parte da equipe, uma boa documentação pode facilitar o trabalho de adaptação ao sistema.

O processo de modelagem, portanto, pode trazer diversas vantagens, ou diversas razões para a sua utilização. Em primeiro lugar, um modelo ajuda no gerenciamento da complexidade inerente à elaboração de um sistema. Em segundo lugar, o modelo permite realizar a comunicação com equipe envolvida: da mesma forma que um engenheiro elabora uma planta e a entrega ao pedreiro para ser executada, um analista de sistemas pode elaborar um diagrama e entregá-lo ao programador para que este realize sua codificação. Como vimos, um modelo também pode facilitar a captura dos requisitos e, dependendo da visão que a equipe possui no momento de planejar o sistema, a modelagem pode ajudar a compreender o futuro do sistema, identificando possíveis funcionalidades que deverão ser implementadas futuramente.

Resumidamente, um modelo pode ser compreendido como uma simplificação da realidade que pode ajudar um profissional, seja ele da área de sistemas ou de qualquer outra área, a entender o problema a ser realizado. Um problema grande e complexo pode ser dividido em pequenos problemas, facilitando o entendimento.

No final da apostila, mais especificamente no apêndice A, você pode encontrar a exemplificação da modelagem usando dois diagramas da UML (linguagem de modelagem unificada) por meio da ferramenta ArgoUML: o diagrama de casos de uso e o diagrama de classes, dois dos mais importantes diagramas.

REFERÊNCIAS

R.S. Pressman, B.R. Maxim, B.R., **Engenharia de Software: Uma Abordagem Profissional**, 8ª edição, Ed. McGraw-Hill, ISBN 9788563308337, 2016.

J.F. Peters, **Engenharia de Software - Teoria e Prática**, ISBN 8535207465, Campus, 2001.