Engenharia de Software

Alguns Fundamentos da Engenharia de Software

O que é Engenharia de Software?

http://www.devmedia.com.br/revista-engenharia-de-software/8028

É a mesma coisa que Ciência da Computação? Ou é uma entre muitas especialidades da Ciência da Computação? Ou dos Sistemas de Informação, ou do Processamento de Dados, ou da Informática, ou da Tecnologia da Informação? Ou é uma especialidade diferente de todas as anteriores?

Na maioria das instituições brasileiras de ensino superior, o conjunto de conhecimentos e técnicas conhecido como Engenharia de Software é ensinado em uma ou duas disciplinas dos cursos que tem os nomes de Ciência da Computação, Informática ou Sistemas de Informação. Raramente, em mais disciplinas, muitas vezes opcionais, e muitas vezes oferecidas apenas em nível de pós-graduação. Algumas instituições oferecem cursos de pós-graduação em Engenharia de Software, geralmente no nível de especialização.

O uso do termo para designar uma carreira profissional também não é muito comum, mesmo em organizações que produzem grande quantidade de software, ou até naquelas em que o desenvolvimento de software é atividade fim. Programas e exames de certificação em Engenharia de Software são pouco conhecidos, ao contrário do que acontece com algumas linguagens e tecnologias usados por esses profissionais.

Em outros países, a situação é um pouco diferente. Algumas universidades americanas oferecem programas de graduação, mestrado e doutorado na área. O IEEE (*Institute of Electrical and Electronics Engineers*), principal organização internacional de profissionais de Engenharia Elétrica, através da Computer Society, que forma o seu ramo em Computação, oferece a qualificação de Profissional Certificado para o Desenvolvimento de Software.

Ciência, Engenharia e Valor

Sem pretender fazer distinções definitivas, vamos explorar o que dizem os dicionários. O Dicionário Aurélio Eletrônico V.2.0 assim define Ciência e Engenharia:

Ciência - Conjunto organizado de conhecimentos relativos a um determinado objeto, especialmente os obtidos mediante a observação, a experiência dos fatos e um método próprio.

Engenharia - Arte de aplicar conhecimentos científicos e empíricos e certas habilitações específicas à criação de estruturas, dispositivos e processos que se utilizam para converter recursos naturais em formas adequadas ao atendimento das necessidades humanas.

Vê-se que, pelas definições acima, a Ciência focaliza acumulação do conhecimento através do método científico, com base em experimentos e observações. Já a Engenharia aplica esses conhecimentos “ao atendimento das necessidades humanas”. Embora o conhecimento seja certamente uma necessidade humana, trata-se de uma entre várias outras, sejam necessidades materiais, como alimentação, moradia, segurança, ou imateriais, como afeição ou autoestima. A tudo aquilo que satisfaz a necessidades, atribui-se um valor. A Engenharia está, portanto, ligada à noção de valor, e a Engenharia de Software busca gerar valor com o processamento de informação. A noção de valor tem muitas consequências práticas importantes, e, de fato, a teoria conhecida como “Engenharia de Software Baseada em Valor” representa uma importante escola de pensamento dentro da área.

Arte, Técnica, Artesanato, Indústria?

Outros termos constantes da definição de Engenharia podem ser explorados de várias formas, com consequências interessantes. Por exemplo, é usada a palavra “Arte”, que o mesmo dicionário define como a

“capacidade que tem o homem de pôr em prática uma ideia, valendo-se da faculdade de dominar a matéria”, ou “a utilização de tal capacidade, com vistas a um resultado que pode ser obtido por meios diferentes”. Na Engenharia de Software, a matéria dominada pelas faculdades humanas consiste em máquinas de processamento da informação, devidamente configuradas e programadas. Nesse sentido, os conceitos de “Arte” e “Técnica” são bem próximos; aliás, a palavra grega *techné* significa, exatamente, Arte.

O termo “Arte” abre outras discussões. Não poucos programadores se consideram como artistas, no sentido de praticantes das Belas Artes, e valorizam critérios estéticos na criação de seus programas. Isso pode ter consequências boas e ruins, do ponto de vista de gerar valor. Por um lado, a busca da elegância pode levar à economia e simplicidade de formas, fazendo com que resultados de melhor qualidade sejam obtidos de maneira mais produtiva. E, principalmente, levando a escrever programas que possam ser mais facilmente reutilizados, mantidos e expandidos. Por outro lado, a autossatisfação do programador pode ter como preço os interesses de quem está pagando pelo trabalho dele, ou de quem o usará. Seja, por exemplo, produzindo programas que ninguém entende, senão o próprio autor (e, depois de certo tempo, nem ele mesmo). Seja, como outro exemplo, introduzindo funções que o autor achou interessantes, mas não são realmente necessárias, nem foram solicitadas.

E muito próxima de “Arte” está a palavra “Artesanato”, que lembra produção caseira, em pequena escala, sem a utilização de métodos industriais, que são caracterizados pela padronização e pela repetição. E, realmente, parece mais difícil aplicar esses métodos industriais na confecção de software, do que nos ramos da engenharia do mundo material. Nestes, as leis físicas impõem limites claramente visíveis ao que pode ser feito. Na Engenharia de Software, a criatividade não é limitada por leis físicas, e sim pela capacidade humana de entender e dominar a complexidade.

Mas não se pode escapar do fato de que a Engenharia de Software tem que resolver muitos problemas de ordem industrial. Raramente é possível construir software profissional sem envolver equipes, às vezes de dezenas ou até centenas de pessoas; raramente é possível trabalhar na área sem a pressão de prazos e orçamentos apertados; frequentemente defeitos de software podem acarretar prejuízos vultosos, e, em certos casos, até riscos à vida humana; muitas vezes empreendimentos de software são afetados por um contexto econômico, político ou social.

Produtos e Ciclos de Vida

A íntima relação entre a Engenharia de Software e a noção de valor significa que essa profissão trata o software como produto. Estão fora do escopo da Engenharia de Software programas que são feitos com objetivo exclusivamente lúdico: a diversão do programador. Estão fora de seu escopo também pequenos programas descartáveis, feitos por alguém apenas para resolver um problema dessa pessoa, e que não serão utilizados por outros. Mas, se um desses programas interessar a outras pessoas, e assim passar a ter valor, aparecerão demandas para melhorar suas qualidades, aumentar suas funções, prolongar sua vida. E aparecerá quem esteja disposto a investir nisso, com a expectativa de ganhar dinheiro. Nesse momento, o programa entrou no escopo da Engenharia de Software e se tornou um produto. Muitos dos produtos de software mais usados seguiram esse caminho.

Todo produto tem usuários: aqueles que efetivamente usam o produto. Alguns produtos são feitos por encomenda de um cliente: aquele que pagará por sua produção. Outros, chamados de produtos de prateleira, são vendidos no mercado aberto, a quem se interessar. Neste caso, quem faz o papel do cliente é quem define que recursos e funções se esperam do produto; talvez um departamento de vendas, ou de marketing, ou de definição de produtos de uma organização, ou até, para produtos menores, os próprios desenvolvedores, colocando o chapéu de investidores de risco. E existem todos os casos intermediários, em que um produto parcialmente pronto é completado, adaptado ou incrementado, por encomenda de um cliente.

Como todo produto industrial, o produto de software tem seu ciclo de vida:

- Ele é concebido para tentar atender a uma necessidade.
- É especificado, quando essas necessidades são traduzidas em requisitos viáveis.

- É desenvolvido, transformando-se em um conjunto formado por código e outros itens, como modelos, documentos e dados.
- Passa por algum procedimento de aceitação e é entregue a um cliente.
- Entra em operação, é usado, e sofre atividades de manutenção, quando necessário.
- É retirado de operação ao final de sua vida útil.

O ciclo de vida compreende muitas atividades, que são assunto das diferentes áreas da Engenharia de Software. A **Figura 1** mostra um modelo do ciclo de vida do software, usando a notação conhecida como UML (*Unified Modeling Language*). Essa notação é usada para descrever muitos aspectos diferentes do desenvolvimento de software, inclusive as sequências de atividades que compõem esse desenvolvimento. O tipo específico de diagrama que é mostrado na figura é o Diagrama de Atividades, que representa uma evolução dos tradicionais fluxogramas, usados pelos programadores há décadas.

É interessante observar que a codificação, que representa a escrita final de um programa em forma inteligível para um computador, é apenas uma pequena parte do ciclo de vida. Muitas pessoas, inclusive alguns profissionais da informática, acham que a codificação é a única tarefa de um engenheiro de software.

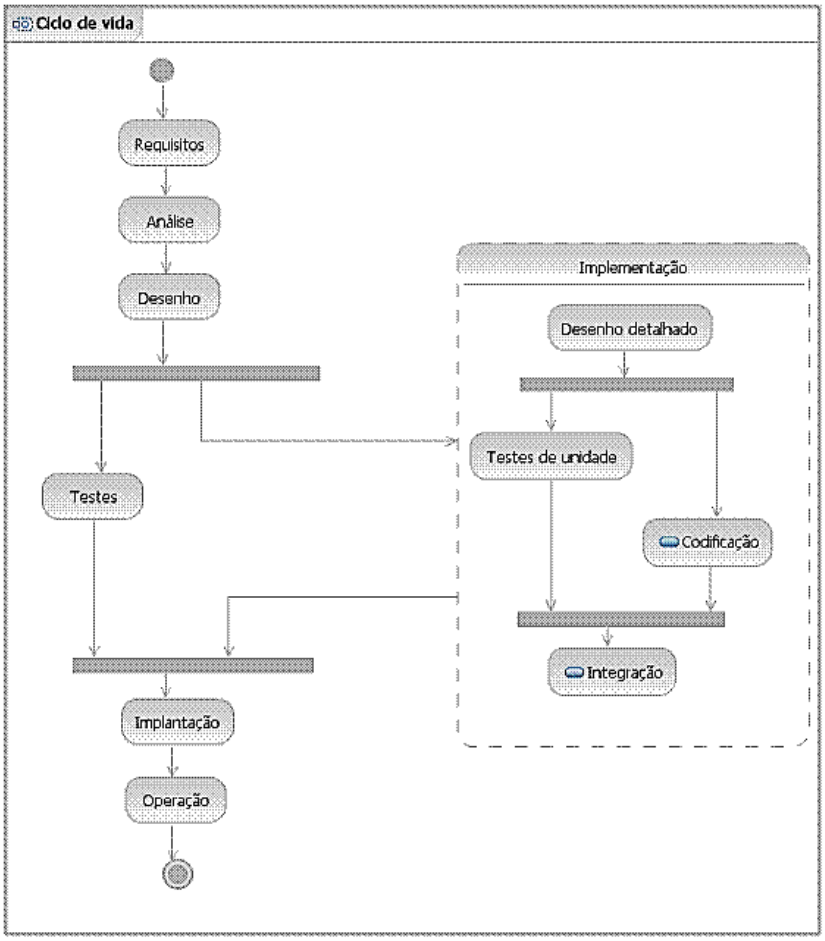

Figura 1 – Modelo UML do ciclo de vida do software

Nos Diagramas de Atividades da UML, a bolinha cheia representa Início; a bolinha com um anel adicional representa fim; cada retângulo simples de cantos arredondados representa uma ação, ou seja, uma atividade simples, da qual não mostram detalhes; cada seta representa uma transição entre atividades; as barras paralelas representam início e término de atividades executadas em paralelo; um retângulo de cantos arredondados com detalhes internos representa uma atividade estruturada, que é composta de outras atividades.

Projetos, Atividades e Estruturas Analíticas

Normalmente, o software é desenvolvido dentro de projetos. Todo projeto tem uma data de início, uma data de fim, uma equipe e outros recursos. O responsável por um projeto é chamado de gerente de projeto. O trabalho realizado dentro de um projeto pode ser descrito por um conjunto de atividades, que podem possuir relações de dependência, paralelismo, e decomposição em atividades mais elementares, como no diagrama da **Figura 1**.

As atividades são delimitadas por marcos, isto é, pontos que representam estados significativos do projeto. Geralmente, os marcos são associados a resultados concretos: documentos, modelos ou porções do produto, que podem fazer parte do conjunto prometido aos clientes, ou ter apenas utilização interna ao projeto. O próprio produto é um resultado concreto associado ao marco de conclusão do projeto, que pode ser utilizado sozinho, ou como componente de um sistema.

O PMI (*Project Management Institute*) é uma organização internacional, com seções em muitos países, inclusive o Brasil, que tem o objetivo de promover e difundir boas práticas de gestão de projetos. Para isso, administra programas de certificação de profissionais nessa área, e publica o guia conhecido PMBOK (*Project Management Body of Knowledge*).

Nesse guia, define-se um projeto como um empreendimento temporário realizado para criar um produto, serviço ou resultado distinto. Um produto, por sua vez, é definido como um objeto produzido, quantificável e que pode ser um item final ou um item componente. Uma atividade é definida como um componente de trabalho realizado durante o andamento de um projeto.

Os relacionamentos entre as atividades que compõem um projeto são mostrados em uma estrutura analítica, que o PMBOK define como uma decomposição hierárquica orientada à entrega do trabalho a ser executado pela equipe do projeto, para atingir os objetivos do projeto e criar as entregas necessárias. Estruturas analíticas de projeto podem ser apresentadas de muitas maneiras. Diagramas de atividade são uma das representações mais usadas atualmente. Outro tipo de representação usual é fornecida por planilhas e cronogramas, como aqueles que são produzidos pela ferramenta Microsoft Project (**Figura 2**).

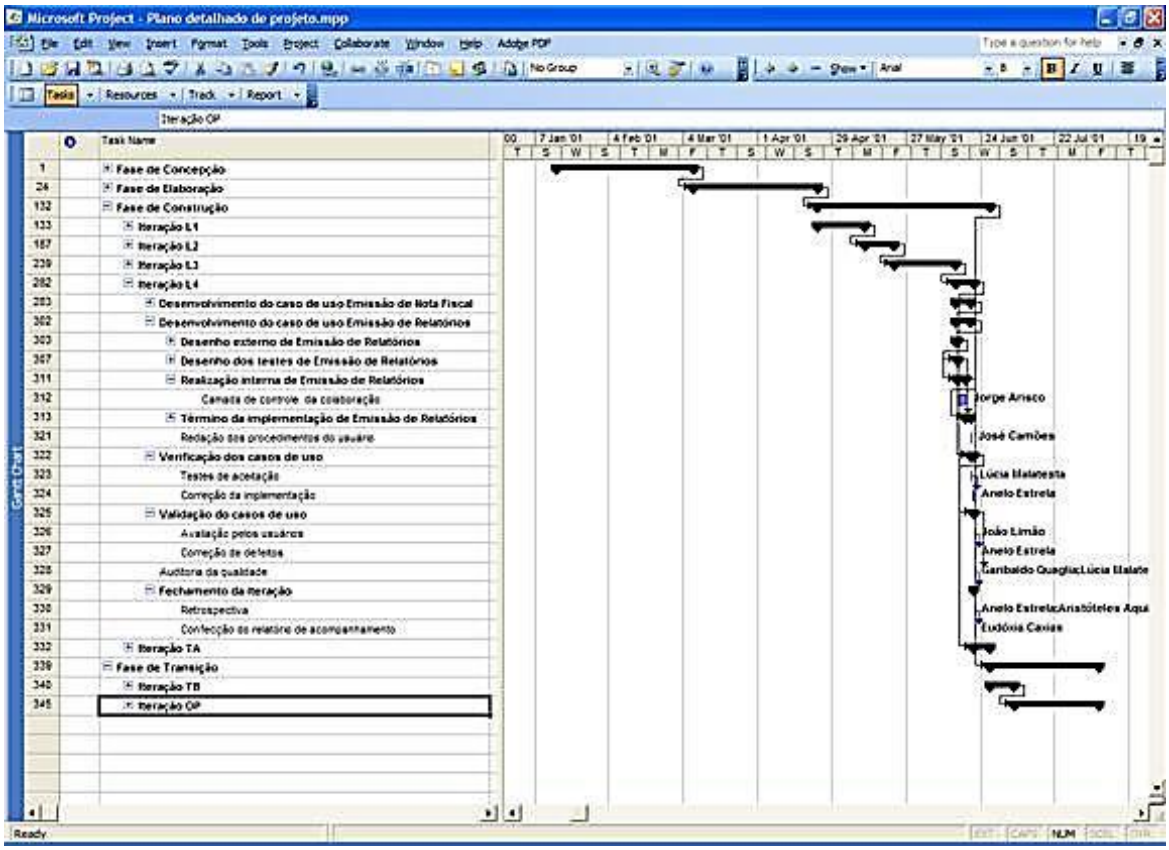

Figura 2 – Cronograma de um projeto

Modelos de Referência e Fatores de Produção

O PMBOK é um exemplo de modelo de referência: uma estrutura de conhecimento que organiza conceitos, práticas e padrões de uma área. No caso do PMBOK, a área focalizada é a gestão de projetos de qualquer natureza, cobrindo assuntos como integração, escopo, tempo, custos, qualidade, recursos humanos, comunicações, riscos e aquisições.

Outro modelo de referência importante na Engenharia de Software é o CMMI (*Capability Maturity Model Integration*), que foi formulado pelo *Software Engineering Institute* da Carnegie-Mellon University. O CMMI foi encomendado e patrocinado pelo Departamento de Defesa americano, popularmente conhecido como Pentágono, que o utiliza para avaliação da capacidade de seus fornecedores de software. Sendo o Pentágono provavelmente a maior organização compradora de software do mundo, o CMMI tem grande aceitação da indústria americana de software, e considerável influência no resto do mundo. A rigor, suas práticas não são restritas à indústria de software, podendo ser aplicadas ao desenvolvimento de outros tipos de produtos.

Os conceitos do CMMI têm raízes em comum com o PMBOK, como se pode observar pela similaridade das definições que adota:

- **Produto** - Resultado que se pretende entregar a um cliente ou usuário.
- **Projeto** - Conjunto gerido de recursos inter-relacionados, que entrega um ou mais produtos a um cliente ou usuário, com início definido e que, tipicamente, opera conforme um plano.
- **Estrutura analítica do projeto** - Um arranjo dos elementos do trabalho e dos relacionamentos deles entre si e com o produto final.

Ao contrário do PMBOK, o CMMI não se limita aos conhecimentos sobre gestão de projetos. Cobre também áreas técnicas, e focaliza principalmente a aplicação dos processos ao desenvolvimento de produtos. Um processo, segundo o IEEE, é uma sequência de passos executados com um determinado objetivo; segundo o PMBOK, é um conjunto de ações e atividades inter-relacionadas, realizadas para obter um conjunto especificado de produtos, resultados ou serviços.

Um projeto representa a execução de um processo, e um processo é uma receita que é seguida durante a realização um projeto; em outras palavras, o projeto é o empreendimento que concretiza uma abstração, que é o processo. Não se deve confundir um processo (digamos, uma receita de bolo) com o respectivo produto (o bolo) ou com a execução do processo através de um projeto (a confecção de um bolo por determinada pessoa, em determinado dia).

Processos, pessoas e tecnologia constituem os fatores de produção, que determinam o grau de sucesso dos projetos, ou seja: se eles conseguem entregar um produto de qualidade suficiente, dentro de um prazo aceitável e com custos viáveis. Portanto, desses fatores dependem a rentabilidade e a sobrevivência das organizações produtoras.

Para muitos profissionais, a tecnologia é o fator mais atraente; muitas vezes, há um otimismo exagerado quanto aos resultados da aplicação de novas tecnologias. Entretanto, tecnologias aparentemente promissoras podem levar para um beco sem saída, e, às vezes, tecnologias consideradas inferiores pelos especialistas lançam raízes permanentes, graças a forças de mercado. Além disso, a tecnologia só se paga quando colocada nas mãos de pessoas capacitadas, trabalhando dentro de processos adequados. Toda introdução de nova tecnologia tem uma curva de aprendizado: as pessoas precisam ser treinadas, cometem inicialmente muitos erros e, por isso, podem até se tornarem menos produtivas, durante algum tempo. Algumas tecnologias mais complexas só se pagam depois de muitos projetos.

Investir na capacitação das pessoas é certamente necessário. Entretanto, formar pessoas é difícil, caro e demorado. Recrutar pessoas capacitadas também: não há sinais de que a oferta de pessoas com alta qualificação no

desenvolvimento de software venha a se igualar à demanda, em futuro próximo. Além disso, muitas pessoas produzem menos que a sua capacidade permitiria, por falta de liderança, por deficiência de ferramentas e suporte, ou por inadequação de processos.

Dos investimentos nos fatores de produção, os investimentos nos processos são geralmente aqueles que podem trazer retorno em prazo mais curto. Processos também não fazem milagres, mas a melhoria dos processos costuma trazer benefícios em prazos relativamente curtos, como é ilustrado por inúmeros relatos de experiência publicados. No mínimo, contribuem para reduzir o desperdício e o retrabalho, o que geralmente já traz ganhos muito significativos.

A melhoria dos três fatores (tecnologias, pessoas e processos) também requer seu próprio processo: ela deve ser feita em etapas bem-definidas e controladas, para que as organizações, em prazos relativamente curtos, possam avaliar se seu investimento está tendo o retorno esperado. E a principal contribuição de modelos de referência como o CMMI é indicar o caminho das pedras e o mapa da mina: onde a experiência coletada indica que o investimento em melhorias é mais rentável, em cada passo da evolução das organizações.

Conclusão

A Engenharia de Software visa à criação de produtos de software que atendam as necessidades de pessoas e instituições e, portanto, tenham valor econômico. Para isso, usa conhecimentos científicos, técnicos e gerenciais, tanto teóricos quanto empíricos. Ela atinge seus objetivos de produzir software com alta qualidade e produtividade quanto é praticada por profissionais treinados e bem informados, utilizando tecnologias adequadas, dentro de processos que tirem proveito tanto da criatividade quando da racionalização do trabalho.

Sites de referência:

- SEI: http://www.sei.cmu.edu/
- CMMI: http://www.sei.cmu.edu/cmmi/
- PMI Brasil: http://www.pmi.org.br/

Wilson De Pádua Paula Filho

Engenheiro Mecânico pelo ITA, Doutor em Engenharia Elétrica pela Escola Politécnica da USP.
Professor Titular aposentado do Departamento de Ciência da Computação da UFMG.
Autor dos Livros “Engenharia de Software: Fundamentos, Métodos e Padrões” e “Multimídia: Conceitos e Aplicações”.