CENTRO PAULA SOUZA

1 INTRODUÇÃO À ENGENHARIA DE SOFTWARE

Prof. Me. Marcelo dos Santos Moreira

Engenharia de Software I

1. INTRODUÇÃO

O desenvolvimento de software é uma atividade de crescente importância na sociedade contemporânea. A utilização de computadores nas mais diversas áreas do conhecimento humano tem gerado uma crescente demanda por soluções computadorizadas.

Para os iniciantes na Ciência de Computação, desenvolver software é, muitas vezes, confundido com programação. Essa confusão inicial pode ser atribuída, parcialmente, pela forma como as pessoas são introduzidas nesta área de conhecimento, começando por desenvolver habilidades de raciocínio lógico, através de programação e estruturas de dados.

Aliás, nada há de errado nessa estratégia. Começamos resolvendo pequenos problemas que gradativamente vão aumentando de complexidade, requerendo maiores conhecimentos e habilidades.

Entretanto, chega-se a um ponto em que, dado o tamanho ou a complexidade do problema que se pretende resolver, essa abordagem individual, centrada na programação não é mais indicada. De fato, ela só é aplicável para resolver pequenos problemas, tais como calcular médias, ordenar conjuntos de dados etc, envolvendo basicamente o projeto de um único algoritmo. Contudo, é insuficiente para problemas grandes e complexos, tais como aqueles tratados na automação bancária, na informatização de portos ou na gestão empresarial. Em tais situações, uma abordagem de engenharia é necessária.

Observando outras áreas, tal como a Engenharia Civil, podemos verificar que situações análogas ocorrem. Por exemplo, para se construir uma casinha de cachorro, não é necessário elaborar um projeto de engenharia civil, com plantas baixa, hidráulica e elétrica, ou mesmo cálculos estruturais. Um bom pedreiro é capaz de resolver o problema a contento. Talvez não seja dada a melhor solução, mas o produto resultante pode atender aos requisitos pré-estabelecidos.

Essa abordagem, contudo, não é viável para a construção de um edifício. Nesse caso, é necessário realizar um estudo aprofundado, incluindo análises de solo, cálculos estruturais etc, seguido de um planejamento da execução da obra e desenvolvimento de modelos (maquetes e plantas de diversas naturezas), até a realização da obra, que deve ocorrer por etapas, tais como fundação, alvenaria, acabamento etc. Ao longo da realização do trabalho, deve-se realizar um acompanhamento para verificar prazos, custos e a qualidade do que se está construindo.

Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software, a qual trata

de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software.

Assim como em outras áreas, em uma abordagem de engenharia de software, inicialmente o problema a ser tratado deve ser analisado e decomposto em partes menores, em uma abordagem “dividir para conquistar”. Para cada uma dessas partes, uma solução deve ser elaborada. Solucionados os sub-problemas isoladamente, é necessário integrar as soluções.

Para tal, uma arquitetura deve ser estabelecida. Para apoiar a resolução de problemas, procedimentos (métodos, técnicas, roteiros etc) devem ser utilizados, bem como ferramentas para parcialmente automatizar o trabalho.

Neste cenário, muitas vezes não é possível conduzir o desenvolvimento de software de maneira individual. Pessoas têm de trabalhar em equipes, o esforço tem de ser planejado, coordenado e acompanhado, bem como a qualidade do que se está produzindo tem de ser sistematicamente avaliada.

1.1. Qualidade de Software

Uma vez que um dos objetivos da Engenharia de Software é melhorar a qualidade dos produtos de software desenvolvidos, uma questão deve ser analisada: **O que é qualidade de software**?

- Se perguntarmos a um usuário, provavelmente, ele dirá que um produto de software é de boa qualidade se ele satisfizer as suas necessidades, sendo fácil de usar, eficiente e confiável. Essa é uma perspectiva externa de observação pelo uso do produto.

- Por outro lado, para um desenvolvedor, um produto de boa qualidade tem de ser fácil de manter, sendo o produto de software observado por uma perspectiva interna.

- Já para um cliente, o produto de software deve agregar valor a seu negócio (qualidade em uso).

Em última instância, podemos perceber que a qualidade é um conceito com múltiplas facetas (perspectivas de usuário, desenvolvedor e cliente) e que envolve diferentes características (por exemplo, usabilidade, confiabilidade, eficiência, manutenibilidade, portabilidade, segurança, produtividade) que devem ser alcançadas em níveis diferentes, dependendo do propósito do software. Por exemplo, um sistema de tráfego aéreo tem de ser muito mais eficiente e confiável do que um editor de textos. Por outro lado, um software

educacional a ser usado por crianças deve primar muito mais pela usabilidade do que um sistema de venda de passagens aéreas a ser operado por agentes de turismo especializados.

O que há de comum nas várias perspectivas discutidas acima é que todas elas estão focadas no produto de software. Ou seja, estamos falando de qualidade do produto.

Entretanto, como garantir que o produto final de software apresenta essas características?

Apenas avaliar se o produto final as apresenta é uma abordagem indesejável para o pessoal de desenvolvimento de software, tendo em vista que a constatação a posteriori de que o software não apresenta a qualidade desejada pode implicar na necessidade de refazer grande parte do trabalho. É necessário, pois, que a qualidade seja incorporada ao produto ao longo de seu processo de desenvolvimento. De fato, a qualidade dos produtos de software depende fortemente da qualidade dos processos usados para desenvolvê-los e mantê-los.

Seguindo uma tendência de outros setores, a qualidade do processo de software tem sido apontada como fundamental para a obtenção da qualidade do produto. Abordagens de qualidade de processo, tal como a série de padrões ISO 9000, sugerem que melhorando a qualidade do processo de software, é possível melhorar a qualidade dos produtos resultantes.

A premissa por detrás dessa afirmativa é a de que processos bem estabelecidos, que incorporam mecanismos sistemáticos para acompanhar o desenvolvimento e avaliar a qualidade, no geral, conduzem a produtos de qualidade. Por exemplo, quando se diz que um fabricante de eletrodomésticos é uma empresa certificada ISO 9001 (uma das normas da série ISO 9000), não se está garantindo que todos os eletrodomésticos por ele produzidos são produtos de qualidade. Mas sim que ele tem um bom processo produtivo, o que deve levar a produtos de qualidade.

Um processo de software, em uma abordagem de Engenharia de Software, envolve diversas atividades que podem ser classificadas quanto ao seu propósito em:

- **Atividades de Desenvolvimento**: são as atividades diretamente relacionadas ao processo de desenvolvimento do software, ou seja, que contribuem diretamente para o desenvolvimento do produto de software a ser entregue ao cliente. São exemplos de atividades de desenvolvimento: especificação e análise de requisitos, projeto e implementação.

- **Atividades de Gerência**: são aquelas relacionadas ao planejamento e acompanhamento gerencial do projeto, tais como realização de estimativas, elaboração de cronogramas, análise dos riscos do projeto etc.

- **Atividades de Garantia da Qualidade**: são aquelas relacionadas com a garantia da qualidade do produto em desenvolvimento e do processo de software utilizado, tais como revisões e inspeções de produtos (intermediários ou finais) do desenvolvimento.

As atividades de desenvolvimento formam a espinha dorsal do desenvolvimento e, de maneira geral, são realizadas segundo uma ordem estabelecida no planejamento. As atividades de gerência e de controle da qualidade são, muitas vezes, ditas atividades de apoio, pois não estão ligadas diretamente à construção do produto final: o software a ser entregue para o cliente, incluindo toda a documentação necessária. Essas atividades, normalmente, são realizadas ao longo de todo o ciclo de vida, sempre que necessário ou em pontos preestabelecidos durante o planejamento, ditos marcos ou pontos de controle. A figura 1.1 mostra a relação entre esses tipos de atividades.

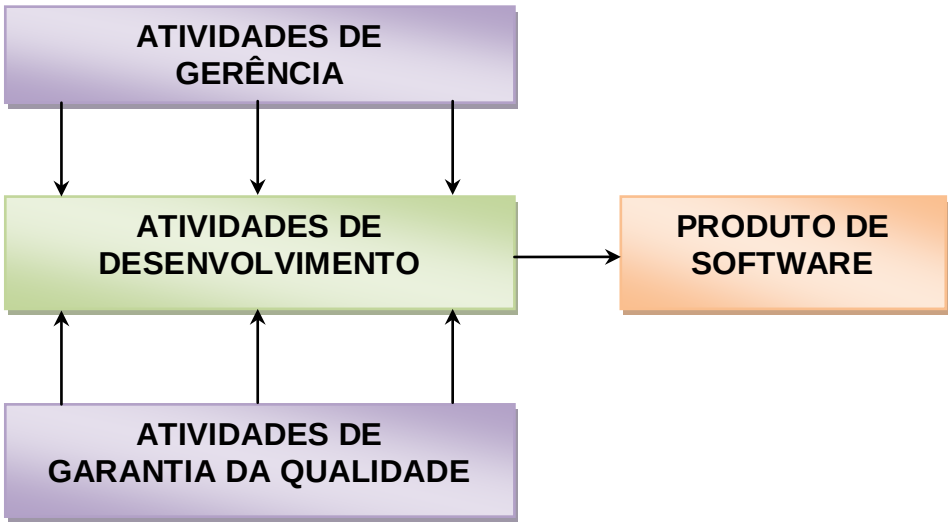

Figura 1.1 – Atividades do Processo de Software

2. PROCESSO DE SOFTWARE

Um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software. Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido.

Os elementos que compõem um processo de software são:

- Processo de Software
  - Processos
    - Atividades
      - Pré-atividades
      - Sub-atividades
      - Artefatos
        - Insumos
        - Produtos
      - Recursos
        - Recursos Humanos
        - Ferramentas de Software
        - Hardware
      - Procedimentos
        - Métodos
        - Técnicas
        - Roteiros

2.1. Definição de Processos

Há vários aspectos a serem considerados na definição de um processo de software. No centro da arquitetura de um processo de desenvolvimento estão as atividades-chave desse processo: **análise e especificação de requisitos**, **projeto**, **implementação** e **testes**, que são a base sobre a qual o processo de desenvolvimento deve ser construído. Entretanto, a definição de um processo envolve a escolha de um modelo de ciclo de vida, o detalhamento (decomposição) de suas macro-atividades, a escolha de métodos, técnicas e roteiros (procedimentos) para a sua realização e a definição de recursos e artefatos necessários e produzidos.

Um processo de software não pode ser definido de forma universal. Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado ao domínio da aplicação e ao projeto específico. Deste modo, processos devem ser definidos caso a caso, considerando-se as especificidades da aplicação, a tecnologia a ser adotada na sua construção, a organização onde o produto será desenvolvido e o grupo de desenvolvimento.

Em suma, o objetivo de se definir um processo de software é favorecer a produção de sistemas de alta qualidade, atingindo as necessidades dos usuários finais, dentro de um cronograma e um orçamento previsíveis.

A escolha de um modelo de ciclo de vida (ou modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida organiza as macro-atividades básicas, estabelecendo precedência e dependência entre as mesmas.

Um modelo de ciclo de vida pode ser entendido como passos ou atividades que devem ser executados durante um projeto. Para a definição completa do processo, a cada atividade, devem ser associados técnicas, ferramentas e critérios de qualidade, entre outros, formando uma base sólida para o desenvolvimento. Adicionalmente, outras atividades tipicamente de cunho gerencial, devem ser definidas, entre elas atividade de gerência e de controle e garantia da qualidade.

De maneira geral, o ciclo de vida de um software envolve as seguintes fases:

- **Planejamento**: O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto.

- **Análise e Especificação de Requisitos**: Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos identificados. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez identificados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo o que o software tem de fazer (e não como fazê-lo).

- **Projeto**: Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Esta arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis de maior detalhamento, até que possam ser codificados e testados.

- **Implementação**: O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada.

- **Testes**: inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado.

- **Entrega e Implantação**: uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação (validação). Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.

- **Operação**: nesta fase, o software é utilizado pelos usuários no ambiente de produção.

- **Manutenção**: Indubitavelmente, o software sofrerá mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases

precedentes é re-aplicada no contexto de um software existente ao invés de um novo.

São fatores que influenciam a definição de um processo: **Tipo de Software** (por exemplo, sistema de informação, sistema de tempo real etc), **Paradigma** (estruturado, orientado a objetos etc), **Domínio da Aplicação**, **Tamanho e Complexidade**, **Características da equipe** etc.

Embora diferentes projetos requeiram processos com características específicas para atender às suas particularidades, é possível estabelecer um conjunto de ativos de processo (sub-processos, atividades, sub-atividades, artefatos, recursos e procedimentos) a ser utilizado na definição de processos de software de uma organização. Essas coleções de ativos de processo de software constituem os chamados processos padrão de desenvolvimento de software. Processos para projetos específicos podem, então, ser definidos a partir da instanciação do processo de software padrão da organização, levando em consideração as suas características particulares. Esses processos instanciados são ditos processos de projeto.

De fato, o modelo de definição de processos baseado em processos padrão pode ser estendido para comportar vários níveis. Primeiro, pode-se definir um processo padrão da organização, contendo os ativos de processo que devem fazer parte de todos os processos de projeto da organização. Esse processo padrão pode ser especializado para agregar novos ativos de processo, considerando aspectos, tais como tecnologias de desenvolvimento, paradigmas ou domínios de aplicação. Assim, obtêm-se processos mais completos, que consideram características da especialização desejada. Por fim, a partir de um processo padrão ou de um processo especializado, é possível instanciar um processo de projeto, que será o processo a ser utilizado em um projeto de software específico. Para definir esse processo devem ser consideradas as particularidades de cada projeto.

Para apoiar a definição de processos, diversas normas e modelos de qualidade de processo de software foram propostas, dentre elas: ISO 9001, ISO/IEC 12207, ISO/IEC 15504, CMM e CMMI. O objetivo dessas normas e modelos de qualidade é apontar características que um bom processo de software tem de apresentar, deixando a organização livre para estruturar essas características segundo sua própria cultura.

Assim, usando essas normas e modelos de qualidade, em uma abordagem de definição de processos em níveis, é possível definir processos para projetos específicos, que levem em consideração as particularidades de cada projeto, sem, no entanto, desconsiderar aspectos importantes para se atingir a qualidade do processo. A figura 2.1 ilustra essa abordagem de definição de processos de software em níveis.

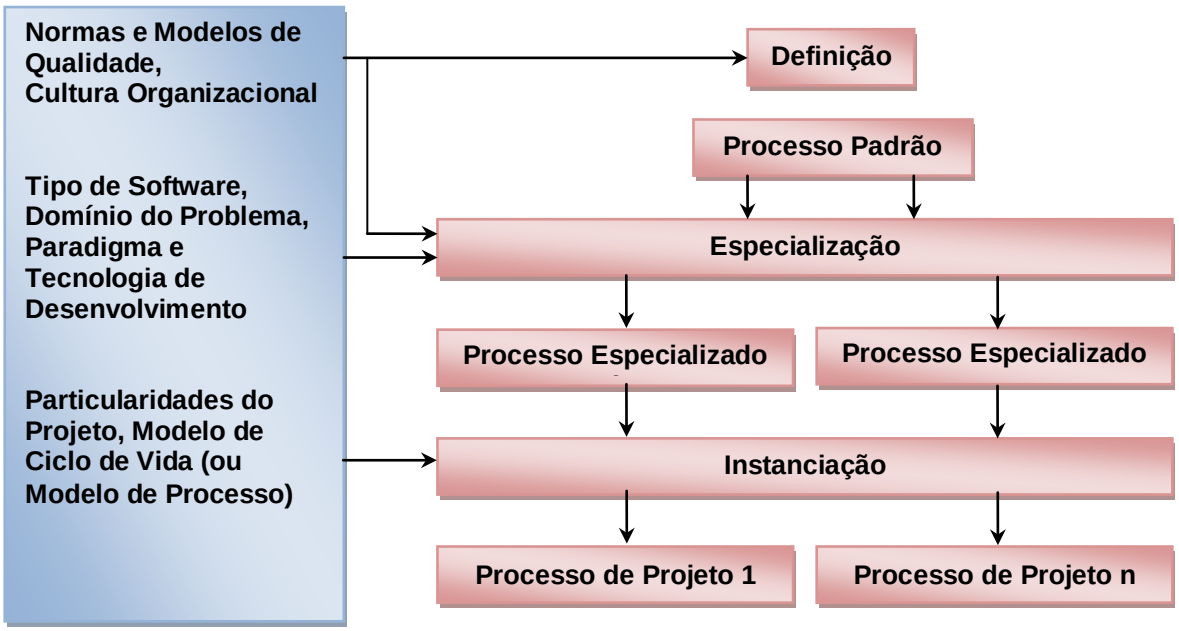

Figura 2.1 – Modelo para Definição de Processos em Níveis

2.2. Automatização do Processo de Software

Com o aumento da complexidade dos processos de software, passou a ser imprescindível o uso de ferramentas e ambientes de apoio à realização de suas atividades, visando, sobretudo, a atingir níveis mais altos de qualidade e produtividade. Ferramentas CASE (*Computer Aided Software Engineering*) passaram, então, a ser utilizadas para apoiar a realização de atividades específicas, tais como planejamento e análise e especificação de requisitos.

Apesar dos benefícios do uso de ferramentas CASE individuais, atualmente, o número e a variedade de ferramentas têm crescido a tal ponto que levou os engenheiros de software a pensarem não apenas em automatizar os seus processos, mas sim em trabalhar com diversas ferramentas que interajam entre si e forneçam suporte a todo ciclo de vida do desenvolvimento, dando origem ao Ambientes de Desenvolvimento de Software (ADSs).

ADSs buscam combinar técnicas, métodos e ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo: gerência, desenvolvimento e controle da qualidade.

Prof. Me. Marcelo dos Santos Moreira

- **Empresário**
- **Consultor de Empresas**
- **Professor Universitário**
- **Mestre em Informática**
  - **Gestão de Sistemas de Informação**
- **Especialista em Sistemas de Informatização Empresarial**
- **Bacharel em Administração**
- **Tecnólogo em Processamento de Dados**

marcelo.moreira@fatec.sp.gov.br