2 Processo

Este capítulo conceitua, de forma genérica, o que é um *processo de desenvolvimento de software* (Seção 2.1), sem detalhar este ou aquele modelo de ciclo de vida. Inicialmente é mostrado que um processo de desenvolvimento, tipicamente, se subdivide em *fases* (Seção 2.2) com objetivos distintos, onde determinadas *disciplinas* (como análise de requisitos, implementação ou teste) podem ser exercitadas com exclusividade ou predominância (Seção 2.3). Processos usualmente são definidos como conjuntos estruturados de *atividades* (Seção 2.4), para as quais são definidos artefatos de *entrada* e *saída*, bem como papeis de *responsáveis* e *participantes*, além de *recursos* necessários. Atividades podem ser detalhadas pela definição de passos de execução, procedimentos e regras. Um exemplo de *documento descritivo* de uma atividade típica de processo é mostrado (Seção 2.5). Finalmente, este capítulo tece alguns comentários sobre a *equipe de processo* (Seção 2.6), que deve ser composta por engenheiros de software de boa capacidade e algumas *normas técnicas* pertinentes (Seção 2.7).

Foi mencionado anteriormente que o desenvolvimento de software deve ser encarado como um processo para que possa ter mais qualidade e ser mais previsível e econômico. Mas o que é um processo? Segundo Sommerville (2003), um *processo* é um conjunto de atividades e resultados associados que produzem um produto de software.

Mas esta definição ainda é muito simples, pois processos usualmente não são apenas conjuntos de atividades, mas atividades estruturadas. Além disso, há mais coisas do que apenas atividades e resultados. Há pessoas, recursos, restrições, padrões a serem seguidos, etc.

A Wikipédia[9] apresenta uma definição mais apropriada: "Um processo de engenharia de software é formado por um conjunto de passos de processo parcialmente ordenados, relacionados com artefatos, pessoas, recursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos".

Um processo, então, pode ser entendido como um conjunto de *atividades*:

a) Interdependentes.
b) Com responsáveis.
c) Com entradas e saídas definidas.

Ao longo deste capítulo, o processo de software será caracterizado de acordo com estas definições e seus detalhes.

Convém, antes de continuar, distinguir os termos *processo*, *projeto* e *modelo* de processo.

Um *projeto* é algo que ocorre em um tempo determinado. Um projeto consiste na execução concreta de um conjunto de atividades que visam à criação de um produto específico.

[9] pt.wikipedia.org/wiki/Processos_de_Engenharia_de_Software (consultado em 26/02/2010)

Um *processo* é um conjunto de regras que definem como um projeto deve ser executado. No jargão da orientação a objetos, o projeto pode então ser considerado como *instância* de um processo. Um processo normalmente é adotado por uma empresa como um conjunto de regras específicas que seus funcionários devem seguir sempre que trabalharem em um projeto. Assim, quando um projeto deve ser planejado na empresa, o responsável deve inicialmente tomar o processo definido e a partir dele definir as atividades concretas, prazos e responsáveis para o projeto específico.

Já o *modelo de processo* é um conjunto de regras mais abstratas que especificam a forma geral de processos. Um modelo de processo apresenta uma filosofia, uma forma geral de comportamento, baseada na qual processos específicos podem ser definidos.

Um modelo de processo para as atividades de projeto e desenvolvimento de software também pode ser chamado de *ciclo de vida*. Assim, quando uma empresa decide adotar um processo, ela deve inicialmente buscar um modelo de processo e adaptar a filosofia e práticas recomendadas para criar seu próprio processo. A partir daí, todos os projetos da empresa deverão seguir este processo definido.

Há várias vantagens em se definir desenvolvimento de software como um processo, entre elas, pode-se citar:

a) *O tempo de treinamento pode ser reduzido*. Com processos bem definidos e documentados é mais fácil encaixar novos indivíduos na equipe do que quando não se tem processos definidos.
b) *Produtos podem ser mais uniformizados*. A existência do processo não garante uma uniformidade na qualidade dos produtos, mas certamente uma equipe com processo bem definido tende a ser mais previsível do que a mesma equipe sem processo algum.
c) *Possibilidade de capitalizar experiências*. Poder-se-ia até imaginar que uma empresa com processo de trabalho bem definido pode estar tolhendo o uso da criatividade por parte dos desenvolvedores. Isso não é verdade, a não ser que a empresa tenha processos engessadores, o que não é bom. Um bom processo, bem gerenciado, deve ter mecanismos para melhoria embutidos. Assim, se um desenvolvedor descobrir um meio de fazer as coisas melhor do que descrito no processo, deve haver meios para incorporar estas alterações no processo.

As seções seguintes vão discorrer sobre a estrutura de processos.

2.1 Fases

Embora a nomenclatura possa variar de um modelo de processo para outro, usualmente se considera que a primeira grande divisão de um processo é em *fases*. Uma fase é um período de tempo no qual determinadas atividades com objetivos bem específicos são realizadas. As *fases* são, então, as grandes divisões dos processos e sua quantidade é normalmente pequena (menos de 10).

Alguns processos como o Modelo Cascata (Seção 3.2) e suas variantes tem fases sequenciais, ou seja, com o passar do tempo o processo de desenvolvimento passa de uma fase a outra, como por exemplo, requisitos, análise, programação, testes e implantação.

Outros modelos podem ter fases cíclicas, ou seja, o desenvolvimento passa repetidamente de uma fase para outra formando um ciclo repetitivo de fases até a finalização do projeto. Exemplos deste tipo de modelo são o Modelo Espiral (Seção 3.8) e o Modelo de Prototipação Evolucionária (Seção 3.9), além dos modelos ágeis (Capítulo 04).

O Processo Unificado (UP - Capítulo 5) é estruturado em quatro fases (embora algumas variantes tenham até seis fases), que são sequenciais no tempo. Mas dentro de cada fase, as atividades são organizadas de forma cíclica. Ou seja, existem ciclos iterativos dentro das fases, mas as fases são sequenciais.

Cada fase de um processo deve ter um macro-objetivo bem estabelecido. No caso do UP, a fase de *concepção* tem como macro-objetivo ter uma primeira abordagem sobre o sistema e seus requisitos. A *elaboração* busca aprofundar a análise e detalhar o *design* do sistema. A *construção* visa produzir código executável e testado. Finalmente, a *transição* visa à instalação e operação do sistema no ambiente final.

2.2 Disciplinas

O termo "disciplina" aparece com mais frequência no Processo Unificado, onde as disciplinas equivalem grosseiramente às fases do Modelo Cascata, embora com outro significado. Disciplina é entendido como um conjunto de atividades ou tarefas correlacionadas, as quais servem a um objetivo específico dentro do processo de desenvolvimento.

Existem, por exemplo, disciplinas de *design* como análise de requisitos, modelagem, programação etc. Mas também existem disciplinas de apoio como gerência de projeto, ambiente e gerência de configuração. Uma implementação do UP (EUP) também apresenta disciplinas relacionadas à empresa como um todo.

As disciplinas usualmente são compostas por tarefas ou atividades que se organizam em um grafo de dependências, que estabelece em que ordem, se for o caso, as atividades devem ser executadas.

No Processo Unificado todas as disciplinas são exercitadas em todas as fases, cada uma com maior ou menor intensidade. Outros ciclos de vida não utilizam explicitamente o conceito de disciplina, sendo organizados normalmente em torno de tarefas ou atividades a serem executadas em cada fase.

2.3 Atividades ou Tarefas

A maioria dos processos de software é organizada em torno de *tarefas* também chamadas às vezes de *atividades*. Toda atividade tem um objetivo principal estabelecido e visa criar ou produzir uma mudança de estado visível em um ou mais artefatos durante a execução de um projeto.

As atividades devem ter *entradas* e *saídas* bem definidas. Uma atividade toma artefatos de entrada e produz como saída novos artefatos e/ou uma modificação bem definida nos artefatos de entrada.

Atividades também devem ter identificados os responsáveis e participantes. *Responsáveis* são as pessoas que devem realizar a atividade e responder pela sua conclusão. Já os *participantes*

agem apenas em resposta à iniciativa dos responsáveis. O ideal é que o responsável seja uma única pessoa (ou cargo). Por exemplo, uma atividade de levantamento de requisitos terá como responsável um analista e como participantes os clientes e usuários.

Cada atividade também necessita de um conjunto de *recursos* alocados, não apenas recursos humanos, que já estarão previstos como responsáveis ou participantes, mas recursos físicos como horas de computador, licenças de software, papel, passagens, etc.

As atividades de um processo normalmente são descritas por um documento. Este documento poderá ter as seguintes seções:

a) Cabeçalho:
    a. Nome do processo.
    b. Nome da fase (se cabível).
    c. Nome da atividade.
    d. Versão do documento.
    e. Responsável (cargo).
    f. Participantes (opcional).
    g. Artefatos de entrada (opcional).
    h. Artefatos de saída.
    i. Recursos alocados (opcional).
b) Corpo, contendo o detalhamento da atividade.

Mais adiante será visto um *template* para o documento de descrição e detalhamento de atividade. Nem sempre as atividades são executadas como uma sequência estrita. Nestes casos pode ser interessante apresentá-las na forma de um algoritmo, ou mesmo de um fluxograma ou diagrama de atividades UML.

Algumas vezes as atividades podem ter descrições distintas dependendo da fase.

2.3.1 Artefatos

Artefatos são quaisquer documentos que puderem ser produzidos durante um projeto de desenvolvimento de software. Artefatos incluem os diagramas, programas, documentos de texto, desenhos, contratos, projetos, planos, etc.

Alguns modelos de processo (como UP) determinam que cada artefato tenha um dono, que é o único que pode modificá-lo ou permitir sua modificação.

Outros modelos (como XP) determinam que artefatos não tenham dono e que possam ser modificados a vontade por qualquer desenvolvedor, desde que exista uma boa razão para isso.

Em qualquer dos casos é importante que todos os artefatos estejam submetidos a um sistema de controle de versão (Capítulo 10) para que eventuais mudanças indevidas possam ser desfeitas.

2.3.2 Responsáveis e Participantes

Os responsáveis por uma atividade são perfis de pessoas ou cargos que respondem pela realização da atividade. Na prática é interessante que qualquer atividade tenha um único

responsável, pois quando existem vários responsáveis pela mesma coisa, pode acontecer que ninguém se sinta realmente responsável.

Quando da descrição de um processo, as atividades devem ser atribuídas a *perfis* ou *cargos* e não a pessoas específicas. Apenas quando o processo for usado em um projeto concreto é que atribuições de atividade a pessoas reais deve acontecer. Por exemplo, uma atividade de análise de requisitos terá como responsável um analista, mas não é o caso de nomeá-lo na descrição da atividade que compõe a documentação do processo. A atribuição de pessoas à atividade é feita durante o planejamento ou gerência de projeto.

Os *participantes* de uma atividade são todas as outras pessoas (seus perfis ou cargos) que precisam participar em alguma atividade para que a atividade seja concluída. Não são necessariamente responsáveis, mas precisam participar. Por exemplo, o cliente deve participar da atividade de levantamento de requisitos, mas o responsável é o analista.

2.3.3 Recursos

Uma atividade, para ser executada, pode demandar recursos. Existem *recursos humanos* e *recursos físicos*. Usualmente os recursos humanos são classificados à parte dos recursos físicos. Recursos humanos são associados às atividades como responsáveis e participantes.

Os recursos físicos, por outro lado, dividem-se em dois grupos: os que são *consumíveis* e os *não consumíveis*.

Não se deve confundir os recursos com as entradas de uma atividade. Entradas são artefatos que servirão como fonte de informação ou que serão transformados na atividade para produzir os artefatos de saída. Já os recursos são ferramentas ou insumos usados na atividade.

Pode-se dizer que as entradas são específicas ao projeto, enquanto os recursos são genéricos. Assim, por exemplo, um *template* de documento é um recurso, mas um diagrama de classes de *design* é uma entrada. Uma ferramenta CASE é um recurso, mas um relatório de *status* de projeto gerado pela ferramenta é uma entrada.

2.3.3.1 Recursos Consumíveis

Recursos consumíveis são aqueles que são gastos quando usados. Por exemplo, folhas de papel, passagens, etc. Esses recursos normalmente são alocados em determinada quantidade.

Por exemplo, para realizar uma reunião do projeto pode-se alocar recursos consumíveis como passagens e diárias para os participantes de outras cidades, folhas de papel pra anotações, biscoitos e café. Nenhum recurso consumível pode ser novamente aproveitado em outra atividade.

2.3.3.2 Recursos Não Consumíveis

Recursos não consumíveis podem ser alocados inúmeras vezes a várias atividades. Porém, normalmente não podem ser alocados a mais de uma atividade de cada vez. Um exemplo de recurso não consumível é o software e o hardware. Enquanto um computador estiver sendo usado em uma atividade não pode ser simultaneamente usado em outra atividade. Mas depois de liberado pode novamente ser usado.

Recursos não consumíveis podem sofrer desgaste e depreciação ao longo do tempo e um dia não serem mais aproveitáveis. Por exemplo, computadores podem para de funcionar, e versões de software podem ficar desatualizadas. Mas isso é gerenciado pelo processo de administração de recursos físicos da empresa como um todo, e normalmente não é dentro de um projeto que se tenta determinar se um recurso não consumível já passou seu tempo de uso. Então, para efeito de processo de desenvolvimento de software, esses recursos podem ser considerados inesgotáveis.

2.4 Detalhamento de Atividades

Atividades podem ser detalhadas por *passos*, complementadas por *procedimentos* e restringidas por *regras*.

2.4.1 Passos

As atividades de um processo podem ser detalhadas em *passos* individuais. Toda atividade necessita de uma descrição, que deve dizer em palavras simples e diretas o que deve ser feito para que a atividade seja realizada.

Basicamente, deve-se informar como cada um dos artefatos de saída são produzidos a partir dos artefatos de entrada. Isso deve ser feito usando um discurso essencial, ou seja, sem entrar em detalhes sobre tecnologias a serem usadas. Detalhes de tecnologia devem ficar restritos aos procedimentos associados à atividade.

2.4.2 Procedimentos

Uma atividade, então, é descrita em termos gerais ou essenciais através de seus passos, que são isentos de tecnologia. Mas, por vezes, pode ser necessário informar como realizar determinados passos com uma tecnologia específica.

Os *procedimentos* representam, então, uma realização tecnológica para o passo essencial definido. Para cada tecnologia possível poderá haver uma descrição de procedimento distinta associada.

O procedimento é uma explicação adicional à atividade, o qual indica como realizá-la com as ferramentas e tecnologia disponíveis. Uma atividade pode ser descrita de maneira semelhante a um caso de uso essencial (Wazlawick, 2011, p. 44), ou seja, mencionando apenas o que deve ser feito sem detalhar a tecnologia a ser usada. Já o procedimento equivale a um caso de uso real, que detalha a atividade com uma tecnologia específica.

Pode haver mais de uma possibilidade tecnológica para realizar uma atividade. Por exemplo, pode haver disponíveis na empresa duas ferramentas CASE. Assim, uma atividade que consiste em realizar modelagem conceitual, por exemplo, terá uma única descrição *independente de ferramenta*. Mas haverá pelo menos dois procedimentos associados à atividade, um para cada ferramenta CASE disponível.

Se houver uma mudança de tecnologia na empresa, pode-se manter os procedimentos antigos para registro e, ao mesmo tempo, acrescentar os novos procedimentos. O interessante é observar que, como a descrição da atividade em si é feita no nível essencial, então ela deve valer para qualquer tecnologia possível.

2.4.3 Regras

A realização de uma atividade pode ainda ser condicionada por *regras* ou *restrições*, que podem se referir a passos, recursos, artefatos, etc. Por exemplo, a atividade de escrever o sumário executivo do projeto pode ter como restrição o fato de que este artefato não deve ter mais do que duas páginas.

Pode-se fazer um paralelo entre atividades e requisitos. As atividades correspondem aos requisitos funcionais (são coisas que devem ser executadas) e as restrições aos não funcionais (a maneira como as coisas devem ser executadas) (Wazlawick, 2011, p. 21).

2.5 *Template* e Exemplo de Documento de Atividade

Esta seção apresenta uma sugestão de *template* para documento descritivo de atividade bem como um exemplo. O *template* é apresentado apenas como referência, sendo que inúmeros outros modelos podem ser usados e seguidos. O *template* é mostrado na Figura 2-1.

Processo:	<nome do processo>
Fase:	<número e nome da fase>
Atividade:	<número e nome da atividade>
Versão:	<histórico de versões do documento>
Responsável: (obrigatório)	<cargo ou pessoa>
Participantes: (opcional)	<cargo ou papel 1> <cargo ou papel 2> ...
Entradas: (opcional)	<artefato 1> <artefato 2> ...
Saídas: (obrigatório)	<artefato 1> <artefato 2> ...
Recursos: (opcional)	<recurso 1> <recurso 2> ...
Passos:	
<passo 1>:	<descrição> <procedimento segundo tecnologia 1> <procedimento segundo tecnologia 2> ... • <regra 1> • <regra 2> ...
<passo 2>:	<descrição> <procedimento segundo tecnologia 1> <procedimento segundo tecnologia 2> ... • <regra 1> • <regra 2> ...
...	...

Figura 2-1: Um *template* de atividade de projeto.

Evidentemente, trata-se aqui apenas de uma proposta. Pode-se discutir, por exemplo, da praticidade de descrever os procedimentos em diferentes tecnologias em diferentes cores. Isso pode ser bastante útil na visualização porque normalmente se está trabalhando com uma tecnologia de cada vez e as informações sobre outras tecnologias são simplesmente inúteis neste caso. Como normalmente há poucas tecnologias possíveis (usualmente apenas uma ou duas), não haverá problemas para distinguir as diferentes cores.

A Figura 2-2 apresenta um exemplo de documento de descrição de atividade preenchido com uma atividade de captura de requisitos em um processo personalizado fictício baseado no Processo Unificado.

Processo:	MP – Meu Processo
Fase:	1. Concepção
Atividade:	1.3 Captura de Requisitos a Partir das Entrevistas
Versão:	1.0 inicial
Responsável:	Analista de requisitos
Participantes:	-
Entradas:	1. Transcrição de entrevistas com o cliente. 2. Sumário executivo do projeto. 3. Definição de escopo do projeto.
Saídas:	1. Documento de requisitos iniciais.
Recursos:	1. *Template* de documento de requisitos. 2. Ferramenta CASE (EA v6.0 ou VP v.8.3).
Passos:	
1.	Listar requisitos funcionais candidatos. EA v6.0: Criar diagrama de requisitos e criar uma caixa para cada requisito candidato preenchendo o texto do requisito no campo "description" VP v8.3: Criar um diagrama de requisitos e uma classe estereotipada como <<requirement>> para cada requisito, preenchendo o texto do requisito no atributo "text", e prenchendo o atributo "kind" com "functional". • Numerar os requisitos funcionais como RF01, RF02, ... • Iniciar sempre com verbo no infinitivo.
2.	Listar requisitos suplementares e não funcionais. EA v6.0: Criar requisitos suplementares em um pacote separado dos funcionais. Indicar os requisitos não funcionais após o texto do requisito funcional associado indicado pela marca "RESTRIÇÕES:". VP v8.3: Criar requisitos suplementares em um pacote separado. Criar requisitos não funcionais como classes estereotipadas do diagrama com atributo "kind" preenchido com o tipo do requisito (interface, segurança, ...) • Associar requisitos não funcionais a algum requisito funcional. • Classificar requisitos suplementares pelo seu tipo: interface, segurança, tolerância a falhas, performance, etc. • Não criar requisitos desnecessários.
3.	Agrupar requisitos funcionais em pacotes. • Não permitir que mais de 20 requisitos estejam em cada pacote, a não ser em casos que se trate efetivamente de requisitos altamente coesos. • Agrupar os requisitos em pacotes por afinidade, ou seja, requisitos mais próximos são aqueles que tratam dos mesmos objetos. • Requisitos do tipo inserir, alterar, remover e consultar, sobre um objeto devem ser agrupados em um único requisito "manter" estereotipado como <<crud>>.
4.	Gerar o documento de requisitos. EA v6.0: Usar o gerador de documentação acessível a partir do menu superior. VP v8.3: Usar a opção "generate report" disponível no meu superior. • Deve ser gerada uma versão pdf para impressão e uma versão html que ficará *online* na intranet do projeto.

Figura 2-2: Um exemplo de atividade descrita de acordo com o *template* apresentado.

Observa-se que no caso dos procedimentos suportados por ferramentas, é importante anotar também a versão da ferramenta para a qual o procedimento foi escrito, visto que de uma versão para outra os procedimentos podem mudar. Cada vez que a ferramenta for atualizada no ambiente de trabalho deve-se revisar se os procedimentos continuam os mesmos e registrar o novo procedimento se for o caso. Alguns modelos de processo, como por exemplo RUP, chamam os procedimentos específicos de ferramentas de "mentores de ferramentas" (Seção 5.3.1.5).

Uma pergunta que pode ser feita por quem ler a descrição de atividade da Figura 2-2 é: "está claro o suficiente?". O descritivo de uma atividade não deve ser detalhado a ponto de ser cansativo para um analista que tenha alguma noção do que está fazendo. Porém, também não pode ser tão genérico a ponto de que dois analistas produzam resultados totalmente

diferentes a partir dele. É necessário que cada passo esteja claramente definido, e quem vai determinar se está claro o suficiente são as pessoas que vão usar esta descrição de atividade.

Se os usuários deste documento acharem que alguma parte não está suficientemente clara, eles devem solicitar mudanças ao engenheiro de software que cuida do processo.

O documento de processo não é estático. Ele vai evoluindo com o passar do tempo e deve ser mantido sob controle de versões (Capítulo 10). Em empresas com maior maturidade, pode-se dizer que ele vai sistematicamente sendo *otimizado*.

Por outro lado, é altamente recomendável que tais documentos sejam elaborados como hipertextos nos quais o leitor possa ler no nível de detalhe que lhe interessa no momento. Por exemplo, apenas os passos das atividades, sem as descrições, ou as descrições sem as regras, ou ainda as descrições, regras e procedimentos referentes a apenas uma tecnologia. Desenvolvedores mais experientes precisarão possivelmente apenas lembrar-se da sequencia de passos e fazer um *checklist* dos artefatos, enquanto que desenvolvedores iniciantes, ou executando processos novos, precisarão de maior detalhamento em relação às atividades.

Uma das ferramentas bastante usada e gratuita para produzir documentos de atividade como hipertextos é o *OpenWiki*[10].

2.6 Equipe de Processo

As organizações devem ter as suas *equipes de processo*, constituídas por um ou mais engenheiros de software que serão responsáveis pela manutenção, avaliação e otimização do processo.

O *Software Engineering Institute*[11] (SEI) publicou um documento sobre como estabelecer equipes de processo de engenharia de software, disponível *online*, que pode ser de grande valia como referência (Fowler & Rifkin, 1990)[12]. Segundo este documento, a equipe de processo é o ponto focal da melhoria de processos em uma empresa. O tamanho do grupo deveria variar entre 1 a 3% do número de profissionais da empresa ligados ao desenvolvimento de software, e ele centraliza e capitaliza o esforço colaborativo dos mais diferentes agentes no sentido da melhoria contínua do processo adotado na empresa.

Organizações muito pequenas poderão ter um funcionário alocado ao processo em tempo parcial.

2.7 Norma NBR ISO/IEC 12207:2008

A norma técnica ISO/IEC 12207:2008 adotada internacionalmente estabelece definições e padrões referentes a vários processos relacionados com a indústria de software.

A norma estabelece processos, atividades e tarefas que devem ser aplicados durante a aquisição, fornecimento, desenvolvimento, operação, manutenção e descarte de software.

[10] www.openwiki.com/ow.asp?OpenWiki
[11] www.sei.cmu.edu
[12] www.sei.cmu.edu/reports/90tr024.pdf

A norma pode ser adquirida no site da ISO[13]. Existe também uma norma, a IEEE 12207 que consiste na adoção e adaptação da ISO 12207 pela IEEE (*Institute of Electrical and Electronic Engineers*). Este documento pode ser adquirido diretamente no site da IEEE[14]. Gray (1999)[15] apresenta um comparativo entre as duas normas.

No Brasil, a ABNT adotou a norma ISO/IEC 12207, transformando-a também em padrão brasileiro, a NBR ISO/IEC 12207:2009. A norma pode ser adquirida no site da ABNT[16].

A ISO/IEC 12207 e suas adaptações apresentam definições e conceitos que são independentes do ciclo de vida escolhido e, portanto podem ser aplicadas a variados contextos.

Para efeito de organização a 12207 divide os processos em quatro grandes famílias:

a) *Processos fundamentais*, que são necessários para que um software seja construído e executado.
b) *Processos de apoio*, que auxiliam outros processos, garantindo qualidade, por exemplo, mas não são fundamentais.
c) *Processos organizacionais*, que são usados no contexto da organização para permitir o melhor acompanhamento e gerenciamento dos projetos.
d) *Processo de adaptação*, onde a norma estabelece como ela própria pode ser aplicada a uma organização ou projeto específico.

Os *processos fundamentais* são os mais fortemente relacionados ao ciclo de vida do software. Esses processos são os seguintes:

a) *Aquisição*. É o processo para obtenção do produto ou serviço relacionado à informática que satisfaça as necessidades da empresa.
b) *Fornecimento*. É o processo cujo objetivo é fornecer um produto ou serviço a terceiros.
c) *Desenvolvimento*. É definido como o processo de transformar um conjunto de requisitos em um produto executável.
d) *Operação*. Possui o objetivo de iniciar e manter o produto operando em seu local definitivo bem como prestar serviços aos usuários.
e) *Manutenção*. Tem como propósito modificar o produto, removendo erros e adequando-o a novos contextos (Capítulo 14).

Os *processos de apoio* têm como objetivo apoiar os processos fundamentais, mas não são eles que compõem as atividades de desenvolvimento propriamente ditas:

a) *Documentação*. Tem como propósito criar e manter informações sobre o produto e o processo de desenvolvimento.
b) *Gerência de configuração*. Tem como propósito gerenciar e manter a consistência entre todas as versões dos produtos do trabalho de forma a manter também sua integridade (Capítulo 10).

[13] www.iso.org/iso/catalogue_detail.htm?csnumber=43447
[14] www.ieee.org
[15] www.abelia.com/docs/122_016.pdf
[16] www.abntcatalogo.com.br/norma.aspx?ID=38643

c) *Garantia de qualidade*. Tem como propósito garantir que os produtos e serviços estejam em conformidade com normas e padrões pré-definidos, bem como consistentes em relação aos requisitos (Capítulo 11).
d) *Verificação*. Tem como objetivo garantir ou confirmar que os produtos refletem os requisitos especificados.
e) *Validação*. Tem como objetivo garantir ou confirmar que os requisitos especificados são os que realmente são desejados pelo cliente.
f) *Revisão conjunta*. Tem como propósito manter um entendimento comum entre os diversos interessados a respeito do produto, do processo ou do serviço.
g) *Auditoria*. Tem como propósito prover uma avaliação independente dos produtos e processos.
h) *Resolução de Problemas*. Tem o propósito de assegurar que todos os problemas levantados sejam resolvidos.

Os *processos organizacionais* garantem o funcionamento da organização:

a) *Gerência*. Tem como objetivo organizar e controlar a realização dos projetos bem como seu desempenho (Capítulo 8.8).
b) *Infraestrutura*. Tem como objetivo manter um ambiente de trabalho adequado.
c) *Melhoria*. Tem como objetivo permitir que os processos possam continuamente ser adaptados visando à otimização do trabalho.
d) *Treinamento* ou *recursos humanos*. Tem como objetivo manter os recursos humanos capacitados para o melhor desempenho possível das suas funções.

O *processo de adaptação* definido pela norma indica como ela pode ser aplicada a diferentes empresas, já que de uma empresa para outra podem variar a cultura organizacional, o modelo de ciclo de vida utilizado no processo de desenvolvimento, e outros fatores.

Pode-se mencionar ainda a norma ISO/IEC 15504, que será discutida em mais detalhes no Capítulo 12, por tratar de qualidade e níveis de capacidade em processo de software. A norma 15504, também conhecida como SPICE, é considerada como uma evolução da ISO/IEC 12207.

