PARTE 1

PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Dentre as áreas do SWEBOK, esta primeira parte do livro aborda "Processo de Engenharia de Software" e parte de "Modelos e Métodos de Engenharia de Software".

Inicialmente, o Capítulo 2 define, de forma genérica, o que se entende por *processo* em Engenharia de Software, o que não deixa de ser semelhante ao conceito de *processo* em outras áreas da Engenharia e da Administração, pois se trata de um conjunto estruturado de atividades, com entradas, saídas, recursos e responsáveis bem definidos.

Os capítulos seguintes apresentam os modelos de processo específicos da indústria de software, iniciando, no Capítulo 3, pelos *modelos prescritivos* (por vezes indevidamente chamados de *clássicos*).

O Capítulo 4 apresenta os principais *modelos ágeis*, que se diferenciam dos prescritivos principalmente por terem seu foco nos fatores humanos em vez das descrições detalhadas de atividades, mais típicas dos processos prescritivos.

O Capítulo 5 apresenta o *processo unificado* e suas implementações. Ele é apresentado como um capítulo à parte, visto que ele incorpora em um único arcabouço muitas das melhores práticas da indústria, inclusive princípios ágeis.

Capítulo 2
Processo

Este capítulo conceitua, de forma genérica, o que é um *processo de desenvolvimento de software*, sem detalhar este ou aquele modelo de ciclo de vida. Inicialmente, é mostrado que, tipicamente, um processo de desenvolvimento se subdivide em *fases* (Seção 2.1) com objetivos distintos, nas quais determinadas *disciplinas* (como análise de requisitos, implementação e teste) podem ser exercitadas com exclusividade ou predominância (Seção 2.2). Processos usualmente são definidos como conjuntos estruturados de *atividades* (Seção 2.3) para as quais são determinados artefatos de *entrada* e *saída*, papéis 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 apresentado na Seção 2.4. Por fim, este capítulo tece alguns comentários sobre a *equipe de processo* (Seção 2.5), que deve ser composta por engenheiros de software de boa capacidade e uma norma técnica que define quais são os *processos típicos da indústria de software* (Seção 2.6).

Já foi mencionado 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), *processo* é um conjunto de atividades e resultados associados que geram um produto de software.

Mas essa definição ainda é muito simples, pois em geral processos não são apenas conjuntos de atividades, mas atividades estruturadas. Além disso, além de atividades e resultados, há mais coisas envolvidas em um processo, como pessoas, recursos, restrições, padrões a serem seguidos etc.

A Wikipédia, possivelmente por ter seus verbetes produzidos pela comunidade de usuários, apresenta uma definição mais completa: "*Um processo de engenharia de software é formado por um conjunto de passos de processo parcialmente ordenados, relacionados com artefatos, pessoas, re-*

cursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos". Um processo, assim, pode ser entendido como um conjunto de atividades:

- Interdependentes.
- Com responsáveis.
- Com entradas e saídas definidas.

Ao longo deste capítulo, o processo de software será caracterizado de acordo com essas definições e seus detalhes. Convém, porém, antes de continuar, distinguir os termos *processo* e *projeto* e a expressão *modelo de processo*.

Projeto é algo que ocorre em um tempo determinado. Consiste na execução concreta de um conjunto de atividades que visam à criação de um produto específico. Assim, um projeto tem um início definido no tempo e um fim determinado. Uma empresa de software pode estar realizando vários projetos ao mesmo tempo.

Processo é um conjunto de regras que definem como um projeto deve ser executado. No jargão da orientação a objetos, o projeto pode ser considerado uma *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 precisa ser planejado na empresa, o responsável deve tomar o processo definido e, a partir dele, definir as atividades concretas, prazos e responsáveis.

Já *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 com base na qual processos específicos podem ser definidos.

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

Há várias vantagens em definir o desenvolvimento de software como um processo, entre elas:

- *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 dispõe deles.
- *Produtos podem ser mais uniformizados*: a existência do processo não garante a uniformidade na qualidade dos produtos, mas certamente uma equipe com um processo bem definido tende a ser mais previsível do que a mesma equipe sem processo algum.
- *Possibilidade de capitalizar experiências*: pode-se imaginar que um processo de trabalho bem definido poderia tolher o uso da criatividade dos desenvolvedores. Contudo, isso não é verdade. Um bom processo deve ter sempre embutidos os mecanismos para sua própria melhoria. Assim, se um desenvolvedor descobrir um meio de fazer as coisas de maneira melhor do que a que está descrita no processo, deve haver meios para incorporar essas alterações em uma nova versão do processo.

As seções seguintes vão discorrer sobre a estrutura de processos, ou seja, suas fases, disciplinas e atividades.

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 a *fase* ou *estágio*. 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 normalmente sua quantidade é pequena (menos de dez).

A norma ISO/IEC TS 24748-1:2016 define que o ciclo de vida do software pode ser concebido em seis fases com objetivos distintos:

- *Concepção*: quando uma primeira exploração das necessidades dos usuários é feita, com a possível construção de modelos ou protótipos de forma a examinar possíveis soluções candidatas.

- *Desenvolvimento*: quando se modela um produto a partir das necessidades identificadas.
- *Produção*: quando o produto é efetivamente construído.
- *Utilização*: quando o produto é colocado em operação no ambiente alvo.
- *Suporte*: quando o produto recebe intervenções de forma a continuar funcionando ou ser aprimorado de acordo com novas necessidades.
- *Desativação*: quando o produto é retirado de uso para possível substituição por outro produto.

Nem todos os modelos de desenvolvimento apresentam todas essas fases tão claramente. Alguns podem ter algumas fases equivalentes a essas, mas com outros nomes. Alguns podem apresentar fases que são a junção de duas ou mais destas ou ainda fases que são a subdivisão de algumas dessas. Outros modelos podem omitir as fases finais, por considerarem que o processo de desenvolvimento acabou quando o produto é entregue. Assim, muitas variações dessa estrutura são encontradas entre os modelos de desenvolvimento de software.

Alguns processos, como o Modelo Cascata (Seção 3.2) e suas variantes, têm fases sequenciais e que se concentram em uma única disciplina. Outros modelos podem ter fases cíclicas, ou seja, o desenvolvimento passa de uma fase para outra, e depois reinicia, formando um ciclo repetitivo de fases até a finalização do projeto. Exemplos desse tipo de modelo cíclico ou iterativo são o Modelo Espiral (Seção 3.8) e o Modelo de Prototipação Evolucionária (Seção 3.9), além de quase todos os modelos ágeis.

O Processo Unificado (Capítulo 5) é estruturado em quatro a seis fases, que são sequenciais no tempo. Mas dentro de cada fase, todas as disciplinas podem ser exercidas com maior ou menor intensidade. Assim, não há, como em outros modelos, fases dedicadas a uma única disciplina, mas uma relação ortogonal entre elas. Adicionalmente, as fases centrais do UP usualmente são organizadas na forma de iterações, ou seja, desenvolvimento cíclico.

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

2.2 Disciplinas

No contexto de processos de desenvolvimento de software, uma *disciplina* é entendida como um conjunto de atividades correlacionadas, as quais servem a um objetivo específico dentro do processo de desenvolvimento. Existem, por exemplo, disciplinas relacionadas à produção, como análise de requisitos, modelagem, programação, teste 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, como gerência de portfólio e reuso estratégico.

As disciplinas usualmente são compostas por 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. Já no modelo Cascata, cada fase exercita apenas uma única disciplina, havendo assim uma coincidência entre o que é *fase* e o que é *disciplina* neste modelo.

2.3 Atividades

A maioria dos processos de software tem como fundamento um conjunto 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 promove 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. Por exemplo, uma atividade de levantamento de requisitos terá como responsável um analista e como participantes os clientes e usuários.

Cada atividade também pode precisar de um conjunto de *recursos* alocados, não apenas recursos humanos, que já estarão previstos na forma de responsáveis ou participantes, mas recursos físicos, como horas de computador, licenças de software, papel, passagens etc. Atividades também podem ser detalhadas em *passos*, complementadas por *procedimentos* e restringidas por *regras*.

As atividades de um processo normalmente são descritas por um documento. Esse documento poderá ter as seguintes seções (um exemplo é apresentado na Seção 2.4):

- Cabeçalho:
    - Nome do processo.
    - Número e nome da fase.
    - Número e nome da disciplina (caso não coincida com a fase).
    - Número e nome da atividade (normalmente um número individual prefixado com o número da fase e da disciplina).
    - Histórico de versões do documento, destacando a versão atual.
    - Responsável (perfil ou papel).
    - Participantes (opcional).
    - Artefatos de entrada (opcional).
    - Artefatos de saída.
    - Recursos alocados (opcional).
- Corpo contendo o detalhamento da atividade através de um conjunto de passos, cada um contendo:
    - Número do passo
    - Descrição
    - Pré-condições (passos da mesma tarefa que já devem ter sido completados)
    - Regras (opcional)
    - Procedimentos (opcional)

Nem sempre as atividades são executadas como uma sequência estrita. As pré-condições referenciadas acima é que vão definir qual a estrutura do grafo de dependência entre os passos. Por exemplo, se os passos 2 e 3 dependem do passo 1 e o passo 4 depende dos passos 2 e 3, então os passos 2 e 3 podem ser executados em paralelo, já que um não depende do outro. Mas para iniciar cada um destes passos é necessário que o passo 1 tenha concluído, e para iniciar o passo 4 é necessário que tanto 2 quanto 3 tenham sido concluídos. Em função dessa estrutura, pode ser interessante apresentar o grafo de dependências entre os passos de uma atividade na forma de um fluxograma, diagrama de atividades UML ou ainda um diagrama BPMN (*Business Process Modeling and Notation*).

Algumas vezes, uma mesma atividade pode ter descrições distintas, dependendo da fase em que é realizada.

2.3.1 ARTEFATOS

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

Alguns modelos de processo (como UP e FDD) 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 à vontade por qualquer desenvolvedor, desde que exista uma boa razão para isso.

Em qualquer um 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.

Convém enfatizar aqui a estreita ligação entre os conceitos de atividade e artefato: toda atividade *deve* produzir um novo artefato ou uma alteração bem definida sobre um artefato existente. Não se admite em processos atividades que nada produzem. Inclusive, quando se está planejando um projeto, deve-se planejar entregas, ou seja, artefatos, e não simplesmente tarefas.

2.3.2 RESPONSÁVEIS E PARTICIPANTES

Os *responsáveis* são perfis de pessoas ou papeis que respondem pela realização de uma ou mais atividades. Na prática, é interessante que qualquer atividade tenha um único responsável. Quando existem vários responsáveis pela mesma atividade, pode ocorrer de ninguém se sentir realmente responsável por ela.

Na descrição de um processo, as atividades devem ser atribuídas a *perfis* ou *papeis*, e não a pessoas. Apenas quando o processo for usado em um projeto concreto é que deve haver atribuições de atividades a pessoas. E neste caso, normalmente, a atividade passa a ser identificada como *tarefa*, já que ela deixa de ser uma descrição abstrata para se tornar uma designação concreta. 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 tarefas concretas a pessoas específicas será feita quando um projeto concreto estiver sendo planejado a partir de um processo existente.

Os *participantes* são todas as outras pessoas (perfis ou papeis) que precisam participar de alguma atividade para que ela seja concluída. Não são necessariamente responsáveis pela atividade, mas precisam participar dela. Por exemplo, o cliente deve participar da atividade de levantamento de requisitos, mas o responsável por essa atividade é o analista.

2.3.3 RECURSOS

Uma atividade, para ser executada, pode demandar recursos. Existem *recursos humanos* e *recursos físicos*. Em geral, os recursos humanos são classificados à parte dos recursos físicos. Os recursos humanos são associados às atividades nos papéis de responsáveis e participantes.

Não se deve confundir os recursos com as entradas de uma atividade. *Entradas* são artefatos que servirão de fonte de informação ou que serão transformados na atividade para produzir os artefatos de saída. Muitas entradas consistem em saídas de atividades anteriores. Assim, elas são específicas do projeto. Já os *recursos* são ferramentas ou insumos usados na atividade. Esses recursos são genéricos, ou seja, eles independem do projeto específico com o qual se está trabalhando.

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

Os recursos físicos se dividem em dois grupos: *consumíveis* e *não consumíveis*.

Recursos consumíveis são aqueles que são gastos quando usados. Por exemplo, folhas de papel, passagens etc. Por exemplo, para realizar uma reunião do projeto, podem-se alocar recursos consumíveis, como passagens e diárias para participantes de outras cidades.

Por outro lado, *recursos não consumíveis* podem ser alocados inúmeras vezes para várias atividades, porém, normalmente não podem ser alocados para mais de uma atividade de cada vez. Exemplos de recursos não consumíveis são o software e o hardware. Por exemplo, enquanto um computador estiver sendo usado em uma atividade, não poderá ser simultaneamente usado em outra. Contudo, depois de liberado, poderá ser usado novamente.

Recursos não consumíveis podem sofrer desgaste e depreciação ao longo do tempo e um dia deixarem de ser aproveitáveis. Por exemplo, computadores podem parar de funcionar e versões de software podem ficar desatualizadas. Mas isso é gerenciado pelo processo de administração de recursos físicos da organização como um todo, e normalmente não é dentro de um projeto que se tenta

determinar se o tempo de uso de um recurso não consumível já expirou. Então, para efeito de processo de desenvolvimento de software, esses recursos podem ser considerados como inesgotáveis.

Por outro lado, recursos consumíveis de uso contínuo como café, toalhas de papel e aluguéis, podem ser alocados ao projeto como um todo e não a cada atividade.

2.3.4 PASSOS

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

Basicamente, deve-se informar como cada um dos artefatos de saída é produzido a partir dos artefatos de entrada. Isso deve ser feito usando-se um discurso *essencial*, ou seja, sem entrar em detalhes quanto às tecnologias a serem usadas. Detalhes de tecnologia devem ficar restritos aos procedimentos associados à atividade. Por exemplo, uma forma de discurso essencial seria dizer "crie um diagrama de classes", já a forma não essencial, usando tecnologia seria "selecione a opção do menu criar/diagrama/classes na ferramenta CASE tal".

Nem sempre os passos de uma atividade são estritamente sequenciais. Por isso, pode ser interessante indicar em cada passo quais as suas pré-condições, ou seja, quais outros passos terão que ter sido executados antes que o passo em questão possa ser iniciado.

2.3.5 PROCEDIMENTOS

Uma atividade 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 poderá haver uma descrição de procedimentos distinta associada.

O procedimento é uma explicação adicional à atividade, indicando como realizá-la com as ferramentas e a tecnologia disponíveis. Uma atividade pode ser descrita de maneira semelhante a um caso de uso essencial (Wazlawick, 2015), ou seja, mencionando-se 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, *independentemente de ferramenta*, mas haverá pelo menos dois procedimentos associados à atividade, um para cada ferramenta CASE disponível.

Se houver mudança de tecnologia na empresa, podem-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.

2.3.6 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 regra o fato de que esse artefato não deve ter mais de duas páginas.

Pode-se fazer um paralelo entre atividades e regras com requisitos funcionais e não funcionais. As atividades correspondem aos requisitos funcionais (são coisas que devem ser executadas) e as regras aos requisitos não funcionais (a maneira como as coisas devem ser executadas) (Wazlawick, 2015).

2.4 Documentação de Processo

Esta seção apresenta uma sugestão de *template* para documento descritivo de atividade e um exemplo. O *template* é apresentado apenas como referência, sendo que inúmeros outros modelos podem ser usados e seguidos. O *template* é mostrado na Tabela 2.1.

Tabela 2.1 *Template* de documentação de atividade de projeto

Processo:	**<nome do processo>**
Fase:	<número e nome da fase>
Disciplina:	<número e nome da disciplina>
Atividade:	**<número e nome da atividade>**
Versão:	**<número da versão atual, data e autor da modificação>** <número da versão, data e autor da modificação> <número da versão, data e autor da modificação> ...
Responsável:	<perfil ou papel>
Participantes: (opcional)	<perfil ou papel 1> <perfil ou papel 2> ...
Entradas: (opcional)	<artefato 1> <artefato 2> ...
Saídas:	<artefato 1> <artefato 2> ...
Recursos: (opcional)	<recurso 1> <recurso 2> ...
Passos:	
<passo 1>:	**<descrição>** Precondições: <números de atividades> Regras: • <regra 1> • <regra 2> ... <procedimento segundo tecnologia 1> <procedimento segundo tecnologia 2> ...
<passo 2>:	**<descrição>** Precondições: <números de atividades> Regras: • <regra 1> • <regra 2> ... <procedimento segundo tecnologia 1> <procedimento segundo tecnologia 2> ...

Evidentemente, trata-se apenas de uma proposta. Pode-se discutir, por exemplo, a praticidade de descrever os procedimentos em diferentes tecnologias usando-se cores variadas. Isso pode ser bastante útil na visualização, porque normalmente trabalha-se com uma tecnologia de cada vez e as informações sobre outras tecnologias são simplesmente inúteis nesse caso. Como em geral há relativamente poucas tecnologias possíveis que chegam a ser usadas, não haverá problemas para distinguir as cores.

A Tabela 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.

Tabela 2.2 Exemplo de atividade documentada

Processo:	**MUP – Meu Processo Unificado**
Fase:	1. Concepção
Disciplina:	2. Requisitos
Atividade:	**1.2.5 Captura de Requisitos a Partir das Entrevistas**.

Versão:	**2.0 13/07/2018 Raul Sidnei Wazlawick** 1.1 05/06/2012 Raul Sidnei Wazlawick 1.0 18/01/2012 Raul Sidnei Wazlawick
Responsável:	Analista de requisitos
Participantes: (opcional)	-
Entradas: (opcional)	• Transcrição de entrevistas com o cliente. • Sumário executivo do projeto. • Definição de escopo do projeto.
Saídas:	• Documento de requisitos iniciais.
Recursos: (opcional)	• *Template* de documento de requisitos. • Ferramenta CASE (EA v6.0 ou VP v.8.3).
Passos:	
1.	**Listar requisitos funcionais candidatos.** Precondições: - Regras: • Numerar os requisitos funcionais como RF01, RF02, ... • Iniciar sempre com verbo no infinitivo. 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".
2.	**Listar requisitos suplementares e não funcionais.** Precondições: 1 Regras: • 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. 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, ...)
3.	**Agrupar requisitos funcionais em pacotes.** Precondições: 1 Regras: • 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.** Precondições: 2, 3 Regras: • Deve ser gerada uma versão pdf para impressão e uma versão html que ficará *online* na intranet do projeto. 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.

Observa-se que, no caso dos procedimentos suportados por ferramentas, é importante anotar também a versão da ferramenta para a qual esse procedimento foi escrito, visto que de uma versão para outra ele pode 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 RUP, chamam os procedimentos específicos de ferramentas de "mentores de ferramentas".

Outra coisa que se pode observar é que este processo hipotético considera que fases e disciplinas são ortogonais, como no caso do UP. Assim, uma atividade pertencente a uma determinada disciplina pode não aparecer em todas as fases ou ainda ter descrições diferentes conforme a fase. Devido a essa ortogonalidade, a disciplina descrita na Tabela 2.2 é numerada como 1.2.5, significando que seria a quinta atividade da disciplina 2 (requisitos) da fase 1 (concepção).

Além disso, observa-se que os passos 2 e 3 na descrição detalhada dependem apenas do passo 1 e, assim, podem ser executados em paralelo ou em qualquer ordem. Já o passo 4 depende desses

dois e, portanto, deve ser executado por último.

Uma pergunta que pode ser feita por quem ler a descrição de atividade da Tabela 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 dois analistas produzirem resultados diferentes a partir dele. É necessário que cada passo esteja claramente definido, e quem vai determinar se isso está claro o suficiente são as pessoas que vão usar essa descrição de atividade.

Se os usuários desse documento acharem que alguma parte não está suficientemente clara, 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 sendo sistematicamente *otimizado*.

De outro lado, é altamente recomendável que tais documentos sejam elaborados como hipertextos que o leitor possa ler no nível de detalhe que lhe interessa no momento. Vários modelos atualmente, como RUP e DAD já são apresentados com tecnologia de hipertexto. Desenvolvedores mais experientes possivelmente precisarão apenas lembrar-se da sequência de passos e fazer um *checklist* dos artefatos, ao passo que desenvolvedores iniciantes, ou que estão executando processos novos, precisarão de maior detalhamento em relação às atividades.

2.5 Equipe de Processo

As organizações devem ter *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* (SEI) publicou um documento, disponível a partir do *QR code* ao lado, sobre como estabelecer equipes de processo de engenharia de software que podem ser de grande valia como referência (Fowler & Rifkin, 1990). [QRC 2.1] Segundo esse documento, a equipe de processo é o ponto focal da melhoria de processos em uma empresa. O tamanho do grupo deve variar entre 1 e 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.

2.6 Processos da Indústria de Software

Pode-se agora perguntar: quais são os processos individuais específicos que devem ser formalmente definidos quando se está estabelecendo um processo de desenvolvimento de software?

Existe uma norma técnica denominada ISO/IEC/IEEE 12207:2017, adotada internacionalmente (inclusive no Brasil, como NBR), a qual estabelece definições e padrões referentes a vários processos relacionados com a indústria de software. Ela estabelece processos, atividades e tarefas que devem ser aplicados durante as seis fases definidas pela norma ISO/IEC TS 24748-1:2016: concepção, desenvolvimento, produção, utilização, suporte e desativação. A norma divide os processos relacionados à indústria de software em quatro grandes grupos:

- *Processos de acordo*, que incluem as atividades de aquisição e fornecimento. Entende-se a aquisição aqui como o processo de estabelecimento de um vínculo entre o fornecedor e o cliente para o início de um projeto de software e o fornecimento como o planejamento do projeto incluindo marcos e entregáveis.
- *Processos organizacionais de viabilização de projetos*, também conhecidos como processos de *suporte*, que dão apoio às atividades de engenharia. Exemplos são os processos de documentação, garantia de qualidade, gerenciamento de recursos humanos, gerenciamento de conhecimento, gerenciamento de portfólio e gerenciamento de infraestrutura.
- *Processos de gerenciamento técnico*, que são os processos de gerenciamento de projeto

mais fortemente relacionados ao produto de software em si. A norma define oito processos de gerenciamento técnico, a maioria dos quais são abordados neste livro:
    - Planejamento de projeto (Capítulo 6);
    - Avaliação e controle de projeto (Capítulo 9);
    - Gerenciamento de decisão;
    - Gerenciamento de risco (Capítulo 8);
    - Gerenciamento de configuração (Capítulo 10);
    - Gerenciamento de informação;
    - Medição (seções 9.5, 11.4 e 13.7);
    - Garantia de qualidade (Capítulos 11 e 12).
- *Processos técnicos*, que são as atividades que historicamente sempre foram associadas à produção de software especificamente. Alguns destes quatorze processos são tratados neste livro enquanto que outros são abordados por Wazlawick (2015):
    - Análise de negócio ou missão (Wazlawick, 2015, Capítulo 2);
    - Necessidades dos interessados e definição de requisitos (Wazlawick, 2015, Capítulo 3);
    - Definição de requisitos de sistema/software (Wazlawick, 2015, Capítulo 5);
    - Definição de arquitetura (Wazlawick, 2015, Capítulo 9);
    - Análise de sistema (Wazlawick, 2015, Capítulos 6, 7 e 8);
    - Implementação (Wazlawick, 2015, Capítulos 10 e 13);
    - Integração (Seção 13.2.2);
    - Verificação (Capítulos 11 e 13);
    - Transição;
    - Validação (Capítulo 13);
    - Operação;
    - Manutenção (Capítulo 14);
    - Desativação.

Todos estes processos definidos pela norma são detalhados por dezenas de atividades que, por sua vez, são detalhadas por centenas de tarefas e entregáveis. A norma não exige que todos os processos, atividades e tarefas sejam executados pela organização. Cabe à organização declarar quais processos, atividades e tarefas ela efetivamente executa e então mostrar que estes processos, atividades e tarefas são executados de acordo com a norma.

REMISSIVO DO CAPÍTULO