Seção 2

**Modelo do processo unificado**

Processo Unificado (PU) surgiu para melhorar o desenvol­vimento de softwares com foco na A/POO (Análise e Projeto Orientados a Objetos). Este modelo de desenvolvimento de software é iterativo e adapta­tivo, permitindo produzir um sistema de grande porte como se fossem vários pequenos sistemas, o que diminui o risco do projeto.

O Processo Unificado foi um marco do desenvolvi­mento de software e foi desenvolvido com o objetivo de garantir a produção de software de alta qualidade.

Nesta segunda etapa da elaboração do projeto para o desenvolvimento do software para a fábrica de bolos do programa “Empreendedores de sucesso” da ONG, você elaborará os casos de uso para a fase inicial da modelagem dos processos envolvidos na fábrica e, com os conhecimentos adquiridos sobre as características do processo unificado (PU), você deverá identificar as fases e atividades de cada processo. Além disso, você poderá aplicar o método interativo e incremental para refinar o desenvolvimento.

Não pode faltar

Um processo de software diz respeito à maneira como produzimos software, ou seja, qual a metodologia, quais técnicas e padrões vamos adotar ao longo do processo. Dentro desse universo de possibilidades, o processo unificado (PU) representa a união de certas metodologias similares que “os três amigos”, Jacobson, Booch e Rumbaugh (2000) justificadamente chamaram de Processo Unificado (PU).

No processo unificado “Um processo define *quem* está fazendo *o quê*, *quando* e *como* alcançar um determinado objetivo” (JACOBSON; BOOCH; RUMBAUGH, 1999 *apud* PRESSMAN, 2011, p. 40, grifos do original).

No PU, os elementos do processo destacados referem-se a:

1. Quem (papel) está fazendo.
2. O quê (artefato).
3. Como (atividade).
4. Quando (disciplina).

Podemos dizer que o modelo PU aperfeiçoou o tradicional processo incremental e o iterativo, eliminando algumas desvantagens dos dois processos.

Por exemplo, no modelo iterativo, às vezes, o software não termina, pois o cliente está sempre solicitando alterações e, se o processo de documentação não for adequado, o software vira uma colcha de retalhos

No caso do incremental, cada parte tem de ser concluída integralmente para se passar à próxima.

Já com o PU proposto por Jacobson, Booch e Rumbaugh (2000) os processos iterativos e incremental caminham em paralelo, permi­tindo que o software vá se tornando robusto a cada refinamento, num caminho para a maturidade do processo.

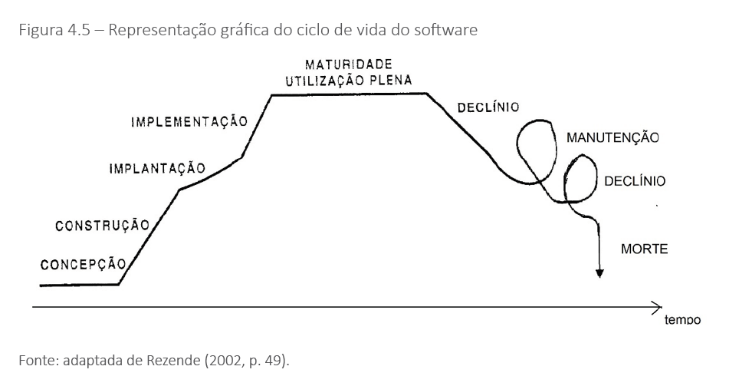
O PU surgiu depois da UML (Unified Modeling Language), que se originou a partir de três métodos (Booch, OMT e OOSE), todos orientados a objeto;

A UML passou a ser utilizada como uma norma pelas empresas de desenvolvimento, mas não era um processo completo para ser seguido no ciclo de desenvolvimento.

A partir dessa necessidade, os mesmos partici­pantes da equipe de desenvolvimento sistematizaram o Processo Unificado Racional (RUP) que é uma especialização, com refinamento detalhado do processo unificado.

O RUP é um processo proprietário, originalmente desen­volvido na empresa RATIONAL SOFTWARE, pelos “três amigos”, e poste­riormente adquirido pela IBM. Em termos gerais, podemos dizer que o PU é um processo sem algumas disciplinas presentes no RUP, porém é de uso público e contempla o ciclo de desenvolvimento do software.

O software, como qualquer produto, tem um ciclo de vida, por exemplo, como ilustrado na Figura 4.5, na qual podemos observar um ciclo com começo (concepção), meio (maturidade e utilização plena) e um fim.



É importante que você não confunda ciclo de vida do produto com modelo do ciclo de vida de desenvolvimento.

Ciclo de vida do produto, segundo Kotler (2018), consiste em 4 fases: introdução (*concepção*), cresci­mento (*desenvolvimento e implantação*)*,* maturidade e declínio. Já modelo do ciclo de vida de desenvolvimento é a arquitetura do processo.

Em engenharia de software existem alguns modelos que podem ser usados para organizar o ciclo de vida de desenvolvimento do software, por exemplo:

Modelo em cascata: 5 fases: (i) análise e levantamento de requisitos, (ii) projeto, (iii) desenvolvimento, (iv) teste e (v) implantação.

Em espiral:

De prototipação:

Incremental: O modelo incremental é entregue em partes (módulos) ao cliente, sendo que cada módulo passa por todas as fases do modelo em cascata.

Iterativo, dentre outros: Já no modelo iterativo, também chamado evolutivo, o cliente parti­cipa ativamente da análise, porém ainda não sabe exatamente especificar todos os requisitos, todas as funcionalidades que o software deve ter ou como as atividades se relacionam.

**CONHECENDO MELHOR O PU:**

Segundo Jacobson, Booch e Rumbaugh (2000), o PU é definido por três aspectos chaves: I. **dirigido por caso de uso**, II. **centrado na arquitetura**, III. **iterativo e incre­mental**.

I – O caso de Uso é o fio condutor do PU – Nesta fase devemos conhecer **o quê** os futuros usuários necessitam e desejam; Devemos nos atentar para que os casos de usos respondam o que cada usuário necessita e não apenas as funções que o sistema precisa ter. Desta maneira o modelo de caso de uso guiará o desenvolvimento no seu projeto (***design***), na imple­mentação e nos testes, avançando através de uma série de fluxo de trabalho (**workflow**).

II. O segundo aspecto, centrado na arquitetura, visa dar ao engenheiro de software uma visão abrangente do sistema, como no caso do projeto de um carro, no qual há o design, a mecânica, o projeto elétrico, aerodinâmico etc. A arquitetura do software também deverá prover ao desenvolvedor estas visões generalizadas, tais como as necessi­dades dos usuários e objetivos estratégicos da empresa.

**Reflita**

O desenvolvimento pelo **ciclo de vida** do processo unificado (PU) é:

DINÂMICO, INCREMENTAL e ITERATIVO.

**Você está aplicando o PU no desenvolvendo de um software, qual aspecto você desenvolverá primeiro?**

Este é um típico caso do “ovo e da galinha”.

Ao pensar no sistema você inicialmente precisa **identificar as caracte­rísticas mais importantes**, deixando os detalhes para depois; por outro lado, **quais as necessidades mais importantes do sistema** para cada usuário e quem participa delas?

Você deve ter em mente que **estes dois aspectos devem se encaixar** e saber que **os casos de uso** de chaves constituem **as funções funda­mentais do sistema.** Segundo Jacobson, Booch e Rumbaugh (2000), 5% a 10% dos casos de uso representam estas funções.

Uma boa dica **é começar a arquitetura pela plataforma** e **compreender os casos de uso mais abrangente**. À medida que os casos de uso se especificam e vão ganhando maturidade, descobrem-se mais detalhes da arquitetura. Este processo continua até que possamos considerar a arquitetura estável. Esse é um exemplo clássico do aspecto **iterativo e incremental em paralelo do PU**.

III. O terceiro aspecto do PU é o iterativo e incremental. A iteração consiste em dividir o projeto em subprojetos menores e o resultado de uma iteração produz um incremento.

Segundo Jacobson, Booch e Rumbaugh (2000, p. 7), “as iterações fazem referência aos passos no fluxo de trabalho e os incrementos ao crescimento do produto”;

O processo iterativo no PU é controlado. Isso reduz os riscos de aumento de custos com incrementos específicos, os prazos são melhor contro­lados e também acelera o ritmo de desenvolvimento, pois reconhece um aspecto ignorado: dificilmente em um sistema complexo, no início, é possível definir totalmente os requisitos e necessidades do usuário.

**Assimile**

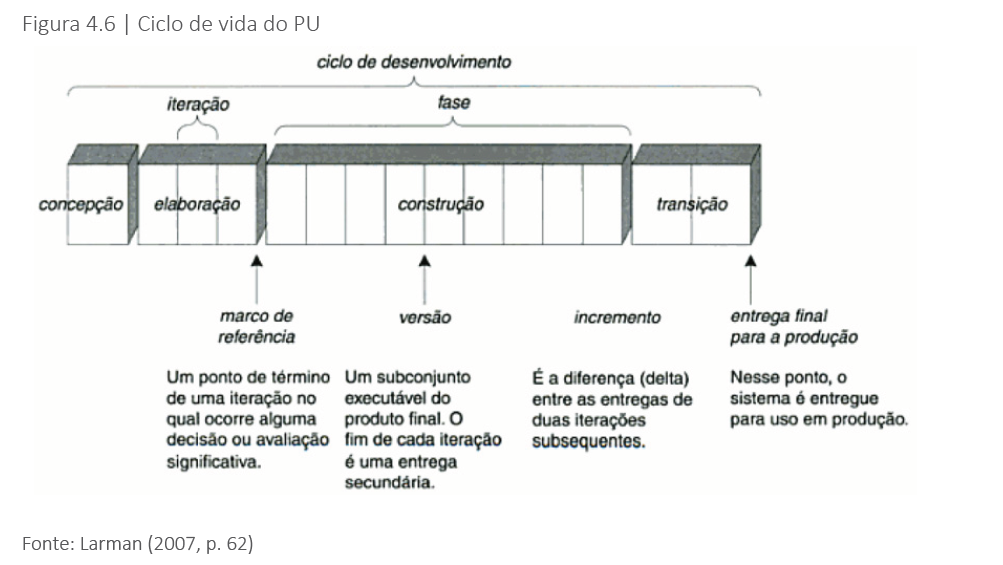
INTERAÇÃO ≠ ITERAÇÃO

INTERAÇÃO = ATO DE INTERAGIR COM OUTROS;

ITERAÇÃO = AÇÕES REPETITIVAS SOBRE CERTAS ATIVIDADES OU OBJETOS; O RESULTADO É UM NOVO OBJETO RESULTADO DA ITERAÇÃO ANTERIOR;

**O ciclo de vida do PU** é uma série de repetições ao longo da vida do sistema, sendo que cada ciclo completo resulta em uma versão do software, por sua vez cada ciclo **é composto por 4 fases:**

1. **1- Concepção**: Definirá a visão geral do projeto, o escopo e os requi­sitos iniciais.
2. **2- Elaboração**: É uma visão mais refinada dos requisitos e da arquite­tura, análise de riscos e estimativas.
3. **3- Construção**: É o momento de desenvolvimento do sistema, começando pelos elementos mais fáceis, e inicia-se a preparação para a implantação.
4. **4- Transição**: É a fase de implantação do sistema, ou seja, a entrega.



O PU descreve um processo por quatro elementos básicos: papel (**quem**), artefato (**o quê**), atividade (**como**) e disciplina (**quando**):

1. **Papel** (*worker*, trabalhador): é a identificação das responsabilidades de cada indivíduo (quem faz o quê), o papel do trabalhador naquele momento, o ator da cena. Ao longo do processo, um *worker* pode ter várias responsabilidades e desenvolver uma série de atividades.
2. **Artefato**: é o termo utilizado para identificar qualquer produto de trabalho, seja código, esquema de banco de dados, diagramas, modelos etc. É o produto que o *worker* gera.

**Atividade**: é a tarefa executada pelo *worker* com o objetivo de produzir ou alterar um artefato.

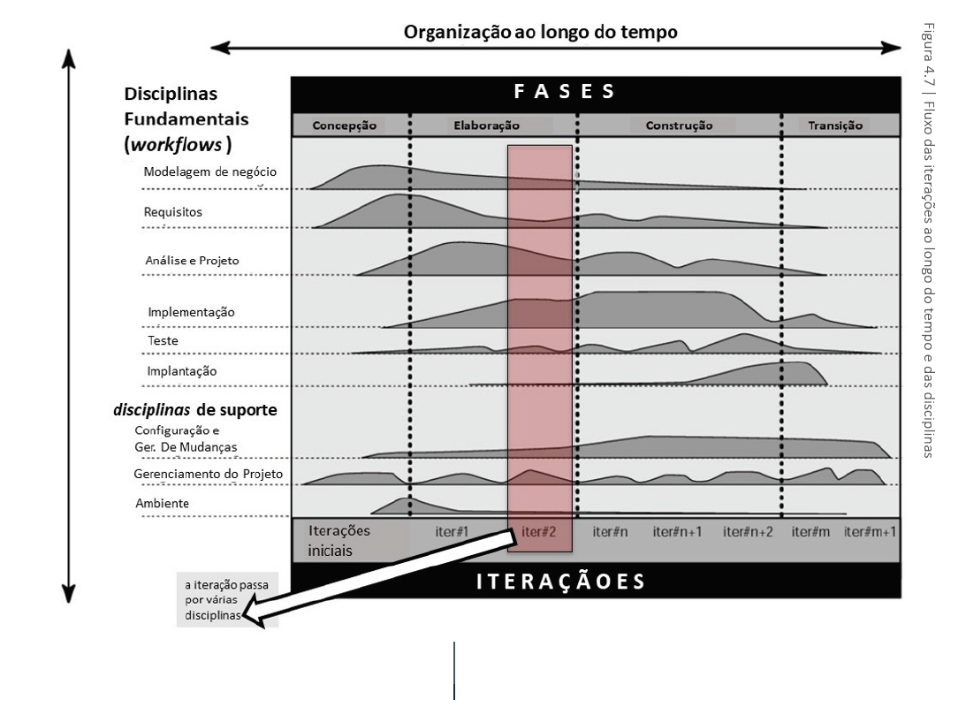
**Disciplina** (*workflow* – fluxo de trabalho): é a sequência de atividades que gera um resultado significativo. Os fluxos estão associados a três perspectivas: dinâmica (tempo), estática (atividades) e boas práticas. Em cada uma encontramos um conjunto de disciplinas.

Segundo Jacobson, Booch e Rumbaugh (2000), as disciplinas funda­mentais, também chamadas fluxo de trabalho do PU são: (i) **modelagem de negócio**, (ii) **requisitos**, (iii) **análise & projeto**, (iv) **implementação**, (v) **teste e implantação**.

Cada ponto de iteração é um marco no PU e finaliza com a entrega de um incremento.

Neste ponto é avaliado se os objetivos foram alcançados e, se necessário, ajustes são realizados.

Embora para cada iteração todas as disciplinas podem estar presentes, o tempo dedicado a cada uma, muda de iteração para iteração. O que é natural no desenvolvimento de um sistema.



Vamos agora analisar o que cada uma destas disciplinas contempla.

**I. Modelagem de negócio**: o objetivo é documentar quais os objetivos de negócio estão envolvidos no processo e são analisados, buscando entender como o negócio deve apoiar os processos associados.

Muitos projetos podem optar por não fazer modelagem de negócios.

Os casos de uso de negócios são utilizados nessa fase.

**II. Requisitos**: Tem por objetivo descrever o que o sistema deve fazer. Tem como base os casos de uso para identificar os requisitos. Nessa fase surge um importante elemento na engenharia de software, os atores. Segundo artigo publicado pela IBM: “Para isso, devem ser identificados os atores envolvidos, todos os quais representam os usuários e qualquer outro sistema que venha a interagir com o sistema em desenvolvimento” (IBM, 2005, p. 4, tradução nossa). O foco está nas funcionalidades do sistema e tanto desenvolvedores como o cliente deverão concordar com a descrição dos requisitos

**III. Análise e projeto:** estas disciplinas geram os modelos do sistema objetivando as implementações futuras. Em conjunto, essas fases melhoram a compreensão dos requisitos funcionais, definido nos casos de uso. Vale lembrar que “os requisitos funcionais são decla­rações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações” (SOMMERVILLE, 2007, p. 80).

Segundo Larman, “a análise enfatiza a investigação do problema e dos requisitos, em vez de uma solução” (LARMAN, 2005, p. 34), ou seja, como será usado e quais serão as suas funções, deixando para a imple­mentação o modo como fazer isso.

O projeto dá ênfase à solução conceitual, ou seja, cria os modelos para satisfazer os requisitos. O projeto é para o desenvolvedor e não para o usuário. Consiste em subsistemas bem definidos, com classes estruturadas em pacotes e deverá representar os componentes a serem implementados.

Vale lembrar que um projeto focado na arquitetura representará uma série de visões do sistema, ou seja, abstrações das características mais importantes, facilitando futuras mudanças quando os requisitos funcionais sofrerem alterações.

**Reflita**

Análise e projetos são úteis quando?

Quando você inicia o desenvolvimento de um sistema, é muito impor­tante ter clareza do problema para o qual você irá criar a solução

**Fazer a coisa certa** está totalmente conectado com entender o problema que queremos resolver, quais são as pessoas e objetos impac­tados por esse problema, como essas pessoas estão tentando resolver atualmente e quanto deste problema queremos resolver.

Esta compreensão gera clareza e isso vem bem antes de criar a solução, esta é a fase da análise. Nesta fase não importa muito como vamos resolver e sim a clareza sobre qual problema vamos resolver. *Logo, comece fazendo a coisa certa: uma boa análise do problema*.

Encontrado a solução certa, a partir da análise correta, será o momento de fazer as coisas corretamente, isto é, o projeto. O projeto envolve conhecimento técnico, processos e metodologias adequadas. Portanto é necessário um domínio adequado destes conhecimentos, preparação para fazer um projeto correto, isto é, fazer certo.

Se você permite questionar constantemente, se **é** a coisa certa a ser feita e aprende nesse processo a buscar resultados, você passa a aceitar que o aprendizado é importante, que você pode mudar de ideia ou adaptar o que está fazendo, pois o que importa é resolver o problema. Essa é uma das propostas do PU iterativo-incremental, reavaliar e evoluir.

Portanto, análise e projeto são úteis, pois permitem saber se:

1. Você fez a coisa certa (*análise*).
2. Se você fez certo a coisa (*projeto*).

**IV. Implementação:** o seu foco está direcionado à construção do software, o desenvolvimento do código, a famosa “mão na massa”. Também prepara os primeiros testes, chamados *beta teste*.

1. **V. Testes**: nesta etapa deve ser descrito quais os procedimentos a serem avaliados.
2. Segundo as recomendações do documento *RUP: boas práticas para o desenvolvimento de software* (IBM,2005), os objetivos do teste são: verificar a interação entre objetos e componentes, verificar se todos os requisitos foram implementados corretamente e identificar e garantir que os defeitos sejam resolvidos antes da implantação do software.
3. **VI. Implantação**: o objetivo do fluxo de trabalho da implantação é entregar o produto com êxito para os usuários, e envolve a aceitação formal do software pelo cliente.
4. Entre as atividades do fluxo de trabalho da implantação temos a distribuição, instalação, migração dos dados e de software já existente.
5. As demais disciplinas estão associadas ao suporte e foram incluídas no Processo Unificado Racional (RUP) para suprir disciplinas não contem­pladas pelo PU, são elas: gerenciamento de mudanças e configurações, geren­ciamento de projeto e ambiente. Basicamente visam a maturidade do projeto.
6. O cuidado a tomar tanto no PU quanto no RUP é a com a atualização da documentação e a definição de marcos precisos para cada iteração.
7. Podemos observar que as diferenças entre o processo unificado racional (RUP) e o PU estão em que o RUP cobre as disciplinas de suporte, voltadas para a maturidade do produto, e é um framework proprietário da IBM nos dias atuais.
8. Você conheceu as fases do PU e suas disciplinas (fluxo de trabalho). Note que esse processo é muito indicado a grandes projetos nos quais há o envol­vimento de vários desenvolvedores na equipe.
9. Exercício:
10. Vamos começar retomando os principais pontos que devem ser seguidos no PU.
11. O caso de uso é o fio condutor do PU e você deverá identificar em cada um, quatro etapas:
12. Quem participa do caso de uso, O PAPEL de cada usuário, suas responsabilidades.
13. Para a fase inicial do projeto da fábrica de bolo, já conseguimos identi­ficar os seguintes papéis de usuários: cliente, fornecedor, vendedor, sistema e-commerce, sistema ERP, funcionário da produção e gerente.

O que será gerado pelo processo, OS ARTEFATOS. São as funciona­lidades do caso de uso.

Também já conseguimos identificar os seguintes artefatos: relatório das entrevistas (DOC\_ENTREVISTAS.docx), receitas dos bolos (DOC\_RECEITAS.docx) e relação das permissões dos usuários versus funções do sistema (DOC\_PERMISSOES.docx).

Como serão realizadas as ATIVIDADES para gerar os artefatos:

Quanto às atividades para gerar os artefatos, podemos identificar: cadas­trar os usuários, cadastrar e manter as receitas de bolo, registrar/associar as permissões dos usuários com as atividades.

Quando serão realizadas as atividades, é a identificação precisa da sequência de passos para realizar cada atividade.

Como esboço inicial, podemos ter a seguinte sequência de atividades: solicitação de compra, separação do produto, expedição do produto.