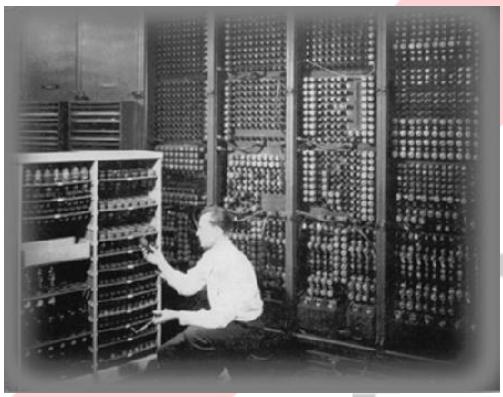
UNIDADE 3 – CONTEXTUALIZAÇÃO DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

MÓDULO 1 – HISTÓRICO DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

01

1 - HISTÓRICO DA CRIAÇÃO DE PROGRAMAS

Até quase o final da década de 1960, existia uma verdadeira corrida em busca do desenvolvimento de computadores cada vez mais velozes. O objetivo da indústria era o de construir processadores mais rápidos e menores, memória e disco com mais capacidade de armazenamento e menores taxas de transferência.



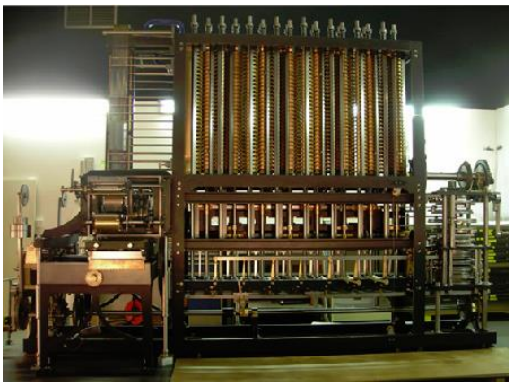
A grande questão é que, com o avanço do "hardware" dos computadores, o universo de funções que poderiam ser realizadas aumentou consideravelmente. Se no início os equipamentos possuíam capacidade extremamente limitada e preços exorbitantes, a expansão verificada na pesquisa e produção de componentes gerou uma significante redução do custo dos computadores.

02

Como já visto em módulos anteriores, os primeiros computadores ocupavam enormes salas, chegando a pesar mais 30 toneladas por conta das suas válvulas eletrônicas e seus quilômetros de fios. Entretanto, a geração posterior, que já utilizava transístores e circuitos impressos, fez que os equipamentos tivessem o tamanho e peso reduzidos em até 10 vezes, ao mesmo tempo em que as capacidades de processamento e armazenamento foram multiplicadas por mil.

Esta evolução da capacidade computacional associada à redução dos preços dos equipamentos foi fundamental para a expansão do uso dos computadores, que deixaram de ter um cunho exclusivamente militar e governamental, e passaram a ser cada vez mais presentes em universidades, centros de pesquisa e em empresas. Foi a demanda gerada por estes novos atores que marcou a virada na história do software.

Na teoria, tem-se que a história do desenvolvimento de programas se iniciou do século XIX, quando a matemática inglesa Ada Lovelace construiu os algoritmos que permitiam a máquina analítica de Charles Babbage processar determinadas funções matemáticas. Esta máquina analítica, entretanto, nunca chegou efetivamente a sair do papel, de modo que os programas não puderam ser efetivamente postos à prova.



03

Os primeiros programas efetivamente executados em computadores eletrônicos só vieram a ser produzidos na década de 40 do século passado. Foi nesta época que foram publicadas as linguagens de programação da primeira geração. A proliferação dos programas só viria, entretanto, na década de 60, com as linguagens da terceira geração, lançadas para permitir o uso apropriado das novas capacidades dos computadores. As inovações que acompanharam o lançamento das linguagens da terceira geração foram fundamentais para que os programadores pudessem atender a expectativa dos novos usuários dos computadores e das novas aplicações dos sistemas de informação.

Falamos da primeira e da terceira geração, entretanto, assim como foi apresentado na evolução dos computadores e uso dos softwares, a história das **linguagens de programação** é dividida em cinco gerações, como apresentado a seguir.

• Primeira geração – linguagens de baixo nível;

• Segunda geração – linguagens assembly;

• Terceira geração – linguagens de alto nível;

• Quarta Geração - linguagens de programação específicas de domínio;

• Quinta Geração – linguagens que dispensam o uso de algoritmo.

Primeira geração
Normalmente chamadas de linguagens de baixo nível, eram pouco intuitivas e intrinsecamente relacionadas à arquitetura da máquina. A programação era inserida diretamente nos terminais dos equipamentos, sem a necessidade de utilização de compiladores para processar as instruções.
A principal vantagem das linguagens de primeira geração é a velocidade de processamento, já que as instruções são processadas diretamente pela CPU, sem a necessidade de compilação ou tradução.
Dentre os exemplos de linguagens de programação da primeira geração, pode-se citar a Microcode.

Segunda geração
O termo foi cunhado para caracterizar as linguagens assembly, já que, apesar de ainda serem caracterizadas como de baixo nível, eram diferentes das linguagens de primeira geração por permitir que o código fosse escrito por um programador, trazendo estrutura lógica para os programas.
Outra característica importante das linguagens desta geração é a necessidade de conversão do código produzido antes que este pudesse ser efetivamente executado no computador.
A principal linguagem de programação desta geração é a Assembly.

Terceira geração
Uma das principais novidades trazidas com esta terceira geração foi a tentativa de tornar as linguagens de programação mais amigáveis para os programadores.
A terceira geração introduziu, ainda, o conceito que ficou conhecido como as linguagens de "alto nível", já que os códigos para programação passaram a estar mais próximos da linguagem humana do que da linguagem de máquina.
A pioneira entre as linguagens de terceira geração foi o FORTRAN (**FOR**mula **TRAN**slator), desenvolvida por um time da IBM com foco em computação científica. As linguagens mais populares do grupo são o "C", "C#" e Java.

Quarta Geração
São linguagens de programação desenhadas para um propósito específico, por este motivo também ficaram conhecidas como linguagens de programação específicas de domínio. Foram pensadas para reduzir o esforço e o custo necessário para o desenvolvimento deste tipo específico de software.
Como exemplo, temos a linguagem SQL, utilizada com o propósito de realizar operações nos mais diversos bancos de dados.

Quinta Geração
Essa linguagem foi pensada para fazer que o computador resolva um problema específico sem a necessidade de se programar um algoritmo para isso, mas apenas lhe fornecendo as restrições. São linguagens frequentemente ligadas a estudos na área de inteligência artificial.
Como exemplo de linguagem de programação de quinta geração, temos o PROLOG (PROgramming LOGic).

04

USO DE DADOS NA TECNOLOGIA DA INFORMAÇÃO

Como foi apresentado, à medida que os equipamentos computacionais começaram a ganhar importância nos mais variados segmentos, sentiu-se a necessidade de aplicações de software mais robustas. Após a década de 60, em que se verificou uma ênfase nos avanços dos equipamentos de hardware e automatização dos processos burocráticos das empresas, observou-se também o início de uma tendência mundial não apenas do armazenamento, mas da manipulação e processamento dos dados originários dos sistemas transacionais, atividade intimamente dependente dos programas de computador.

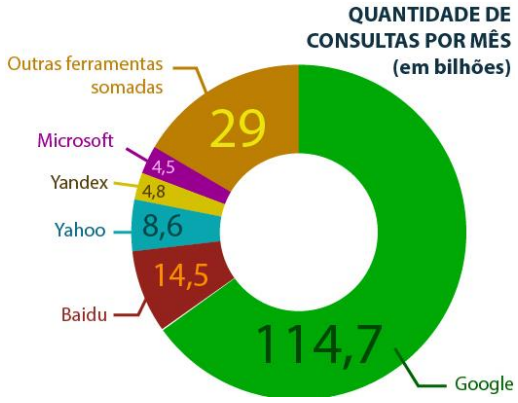
Este período de intensificação do uso de dados e rápido crescimento da demanda por software foi marcado por fatos positivos, como o aparecimento dos sistemas de gerenciamento de bancos de dados (SGBD), e negativos, como a crise do software. Os bancos de dados, como visto na unidade anterior, são categorias de aplicações construídas especificamente para otimizar o processo de armazenamento e manipulação de grandes volumes de dados. Ainda hoje esta é uma das aplicações de software mais utilizadas em todos os segmentos. Já a crise do software foi um período que ficou caracterizado pela ocorrência de projetos de desenvolvimento de software que geraram produtos de baixa qualidade, com estouro do orçamento e atraso considerável nos prazos de entrega.

A **crise do software** emergiu por conta de um aumento exponencial da complexidade das soluções e da demanda por software, associada à falta de experiência das empresas na construção de sistemas complexos. Esta crise, que abateu o mundo no início dos anos 70, teve como consequência um aumento na pesquisa e desenvolvimento em Engenharia de Software, que desencadeou, nas décadas subsequentes, a criação de processos e metodologias que levaram a uma considerável melhoria no gerenciamento da qualidade do software.

A crise do software será um assunto abordado com detalhes em outro Módulo desta Unidade.

05

Dois dos maiores exemplos do uso de dados na tecnologia da Informação são Google, Twitter e Facebook. Para se ter uma ideia, o Google realiza hoje mais de 40.000 consultas por segundo em sua base de páginas web. Isso representa mais de 3,5 bilhões de buscas por dia e cerca de 1,2 trilhão de buscas por ano, o que coloca o buscador como o site mais referenciado da categoria.



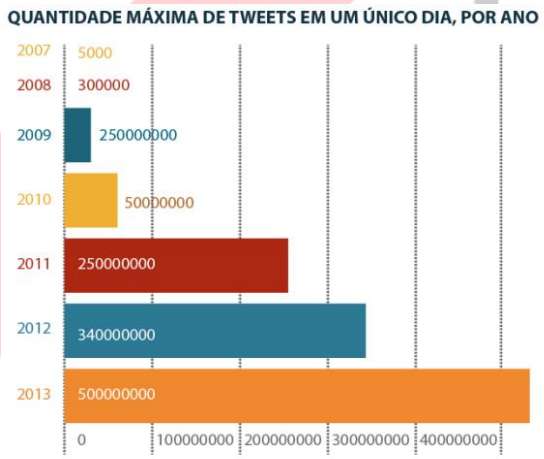
É sempre importante reiterar o fato de que todo esse avanço no uso massivo de dados é consequência do avanço do hardware dos computadores. Em 1999, a operação de processamento e construção do índice de 50 milhões de páginas era executado pelo Google em 1 mês. Hoje, esta mesma operação é realizada em menos de 1 minuto, uma redução de 43.200 vezes do tempo de processamento em apenas 15 anos.

Saiba+

O maior buscador do planeta é cercado de curiosidades. Você sabia que cada consulta realizada ao Google percorre mais de 2500 quilômetros até um centro de processamento e dados e na volta do dado para o usuário? Veja esta e outras curiosidades em http://www.internetlivestats.com/google-search-statistics/#trend

06

Você consegue imaginar que no tempo que você está gastando para ler esta linha quase 20 mil tweets já foram postados no Twitter? Em média, são cerca de 6 mil postagens a cada segundo, o que corresponde a 350 mil por minuto, 500 milhões por dia e 200 bilhões por ano, o que faz do Twitter a principal ferramenta online de postagem de mensagens curtas. Você consegue imaginar a capacidade computacional para armazenar e processar volume tão grande de dados?



O último exemplo abordado do uso de dados na tecnologia da informação é o Facebook. Por todo o mundo, são mais de 1,2 bilhões de usuários ativos desta rede social, os quais produzem mais de 4,5 bilhões de “likes”, trocam mais de 10 bilhões de mensagens e postam mais de 300 milhões de fotos por dia, gerando, apenas em imagens, aproximadamente 420 Terabytes de dados por mês. Toda essa arquitetura só funciona com base em uma estrutura computacional robusta e um conjunto de softwares confiáveis e apropriados para o tratamento de grandes volumes de dados.

07

2 - Conceito de Software

O software, ou programa de computador, é uma sequência de instruções ou comandos lógicos que são executados ou interpretados por um computador com o intuito de executar uma tarefa específica.

A primeira ocorrência do termo “software” na literatura foi identificada no artigo intitulado "The Teaching of Concrete Mathematics”, publicado pelo professor John W. Tukey em 1958.

Os softwares são comumente classificados em três diferentes tipos:

- aplicativos,
- softwares de sistema e
- maliciosos.

Vejamos cada um deles a seguir.

The Teaching of Concrete Mathematics

Para ler o artigo original, em inglês, na íntegra, acesse:
http://www.jstor.org/discover/10.2307/2310294?uid=3737664&uid=2&uid=4&sid=21104490760627

08

Os **aplicativos** são os tipos de software mais conhecidos, pois tem como principal objetivo auxiliar diretamente os usuários na execução de tarefas específicas. É nesta categoria que estão os pacotes de escritório, com os editores de texto e planilhas eletrônicas, os navegadores de internet e as ferramentas de mídia e entretenimento.

Já os **softwares de sistema** são desenhados para prover o ambiente necessário para o correto funcionamento dos aplicativos. São os responsáveis por manter a interface direta com o hardware computacional, abstraindo parte da complexidade desta operação. Nesta categoria se encontram os sistemas operacionais, como o Microsoft Windows, as distribuições Linux e Unix, e os drivers de dispositivos, que permitem o correto funcionamento de componentes como as placas de rede, de vídeo e de áudio.

A última categoria é composta pelos códigos **maliciosos**, ou “malware”. São pragas computacionais desenvolvidas para provocar danos e, muitas vezes, auxiliar na prática de crimes. Como exemplos, podemos citar os vírus, cavalos de troia, os “spywares” e os “worms”.

Algumas das principais infecções causadas por malwares tiveram efeitos devastadores sobre usuários de todo o mundo, principalmente grandes corporações. Clique aqui e conheça alguns dos principais malwares da história.

Conheça alguns dos principais malwares da história

Storm worm – utilizou-se da ocorrência de um desastre natural para levar os usuários a abrir o anexo contaminado do e-mail, cujo título era – 230 mortos em temporal na Europa (“230 dead as storm batters Europe”). Os computadores infectados se transformavam em zumbis e passavam a fazer parte de uma grande rede de máquinas escravas. Estimou-se entre 1 e 10 milhões a quantidade de computadores atingidos.
Love Bug – o modo de difusão foi similar ao storm worm, através do envio de e-mails. A diferença principal era no assunto do email, que tinha por título o texto “I love you”. Infectou mais de 50 milhões de computadores em apenas nove dias, causando prejuízos de bilhões de dólares.
Stuxnet – vírus construído especificamente para atacar o software de automação industrial SCADA, produzido pela Siemens. Existe uma corrente que acredita que foi desenvolvido com o objetivo de causar uma pane nas centrífugas nucleares iranianas, fazendo-as girar fora de controle, mas mantendo a aparência de que nada anormal estava acontecendo.

09

3 - Paradigmas de Desenvolvimento de Software

Segundo o dicionário Priberam da língua portuguesa:

Paradigma é algo que serve de exemplo geral ou de modelo; conjunto das formas que servem de modelo de derivação ou de flexão; conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.

Nesta linha, um paradigma de desenvolvimento de software fornece um exemplo geral, um modelo, um conjunto de elementos que caracterizam uma determinada forma de se construir um programa de computador, estabelecendo fronteiras ou limites e determinando como o padrão deve ser seguido.

Dentre os paradigmas existentes, podemos citar:

- Paradigma da Orientação a Aspectos;
- Paradigma da Orientação a Eventos;
- Paradigma Lógico;

- Paradigma da Orientação a Objetos;
- Paradigma Imperativo.

Detalharemos, a seguir, alguns dos mais importantes.

Paradigma é algo que serve de exemplo geral ou de modelo; conjunto das formas que servem de modelo de derivação ou de flexão; conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.

Priberam
Paradigma, in Dicionário Priberam da Língua Portuguesa [em linha], 2008-2013, http://www.priberam.pt/dlpo/paradigma [consultado em 10-08-2014].

10

Paradigma da Orientação a Aspectos

Tem como característica a separação dos níveis de preocupação durante o desenvolvimento de software, organizando o código de acordo com a sua importância. Desta forma, o problema é fragmentado e cada elemento do sistema pensado separadamente, sendo que cada parte passa a ser um aspecto do sistema que deve ser tratado isoladamente, como é exibido na Figura abaixo:

11

Paradigma da Orientação a Eventos

É marcado pela estratégia de controle do fluxo do programa. Enquanto a maioria dos paradigmas tradicionais segue um fluxo padrão, o fluxo do paradigma orientado a eventos é caracterizado pelas chamadas a partir de eventos externos. A sua principal aplicação se dá na construção de interfaces voltadas para o usuário.

Com a ampliação do uso da internet, este paradigma tem-se tornado uma das novas vedetes no cenário da programação de aplicações web, na qual os programas respondem a partir de eventos disparados pelos usuários, como o click em um botão, a seleção de um elemento em uma caixa de listagem ou de um item no menu.

12

Paradigma Lógico

Baseia-se fortemente na utilização da lógica matemática, se relacionando com as linguagens de programação da quinta geração e os estudos na área de inteligência artificial. A própria origem do paradigma está relacionada a criação da linguagem de programação Prolog, no início da década de 70.

Costuma ser muito utilizado na criação de sistemas “especialistas” ou “inteligentes”, que são aqueles que têm por objetivo simular o raciocínio de profissionais humanos com domínio do conhecimento em áreas especificas, como, por exemplo, um sistema que auxilia um médico no diagnóstico de doenças a partir da inserção dos sintomas do paciente.

13

Paradigma da Orientação a Objetos

Neste paradigma, o sistema é modelado com base em objetos que representam elementos do mundo real. Imagine, por exemplo, que o objetivo é construir um programa para gerenciar transações bancárias. Neste sentido, de acordo com este paradigma, seria conveniente pensar em elementos reais, como o correntista, o banco e a conta corrente. Seria intuitivo, ainda, pensar em atributos e principais operações de cada elemento, como o nome e o CPF do correntista, e as operações de efetuar um saque ou um depósito na conta corrente.

A figura abaixo exibe parte da modelagem orientada a objeto para um sistema bancário. Percebam a similaridade entre o modelo e os elementos encontrados no mundo real.

14

PARADIGMA IMPERATIVO

Também conhecido como sinônimo da programação procedural, tem como base a ideia de estados e de operações que modificam estes estados. Sofre forte influência do modelo arquitetural de Von Neumann, já que traz muitas similaridades com o modo de funcionamento do computador. Tem como peças chave a existência de variáveis, de operações de atribuição, a execução sequencial dos comandos e a forma iterativa de repetição.

Como vantagens, destacam-se: a eficiência do modelo, o fato de ser largamente utilizado (a maioria das linguagens de programação utilizam este paradigma) e de ser de fácil entendimento. Como desvantagens, apontamos o relacionamento indireto com as operações de Entrada/Saída e o fato de que este paradigma dá ênfase no "como" uma determinada ação deve ser executada ao invés de enfatizar "o que deve ser feito".

15

4 - A CONSTRUÇÃO DE UM SOFTWARE

A construção de um software é composta por um conjunto ordenado de atividades, muitas das quais já padronizadas pelo mercado. De uma forma simplificada, podemos subdividir este processo em cinco grandes atividades, o levantamento e análise de requisitos, o desenho ou projeto do software, a etapa de implementação e testes de unidade, integração e testes do sistema e operação e manutenção. Detalhes de cada uma destas etapas já foram apresentados nos módulos anteriores.

Apesar das fases do desenvolvimento serem similares nas metodologias das mais diversas instituições, cada empresa tem suas próprias práticas, métodos e tecnologias para construção de software. Mas se cada um age de uma forma, como saber se o processo está adequado? Como avaliar se a forma de "construir" software da empresa está de acordo com as melhores práticas? Para resolver esta questão, surgiram os modelos de maturidade de software.

Os modelos de maturidade de software trazem um conjunto de elementos que auxiliam na avaliação e diagnóstico do nível de "experiência" de uma empresa na criação de sistemas, perpassando todas as atividades do ciclo de desenvolvimento.

16

O *Capability Maturity Model Integration*, ou CMMI, é o mais conhecido modelo de maturidade de software do mercado. Foi criado na década de 80 pela Universidade Carnegie Mellon para ser utilizado pelo Departamento de Defesa norte-americano. O CMMI é dividido em cinco níveis de maturidade e aborda 22 áreas chave do processo. A medida que a expertise aumenta em cada uma destas áreas chave, a empresa vai avançando os degraus do nível de maturidade.

De acordo com o CMMI, toda e qualquer empresa inicia pelo primeiro nível de maturidade, chamado de **Inicial**. Neste estágio, considera-se que a organização ainda possui um processo ad hoc, ou seja, o método de construção é pensado para resolver apenas um problema específico, não existindo uma definição ou documentação padrão que é utilizada para atender a todos os projetos.

O segundo nível de maturidade do modelo é chamado de **Gerenciado**. Nesta categoria, apesar de não ser obrigatório que o processo esteja escrito, ele é monitorado e controlado. Os projetos são planejados de acordo com uma política estabelecida e há garantia de que os membros da equipe de desenvolvimento possuam capacitação suficiente para entregar os produtos com a qualidade desejada dentro dos prazos esperados.

O nível seguinte de desenvolvimento é o **Definido**. Aqui todos os processos, procedimentos, técnicas e padrões são formalmente "escritos". Isto faz com que se consiga manter a consistência na construção de sistemas por toda a organização. Quando o processo, já definido, passa a ser **Gerenciado Quantitativamente**, a organização passa a próxima etapa evolutiva do modelo de maturidade do CMMI. Objetivos quantitativos são índices estatísticos para mensuração da eficiência e eficácia do processo de criação dos sistemas.

Quando a empresa consegue, com base nos índices de medição coletados, trabalhar na melhoria contínua dos seus processos, ela finalmente alcança o último estágio de maturidade estabelecido no CMMI, o nível **Otimizado**.

17

Além de passar a atender as características gerais de cada um dos níveis de maturidade, para avançar entre os estágios, a organização também precisa evoluir na sua metodologia de desenvolvimento de sistemas. Para isso, o CMMI define 22 áreas chave vinculadas a práticas da Engenharia de Software. A distribuição das áreas em cada nível do processo é apresentada na figura abaixo. É importante ressaltar que o nível Inicial não tem qualquer exigência de processos, por este motivo não está representado na figura.

NÍVEIS DE MATURIDADE DO CMMI-DEV v1.3

NÍVEL 2 - GERENCIADO	NÍVEL 3 - DEFINIDO	NÍVEL 4 - GERENCIADO QUANTITATIVAMENTE	NÍVEL 5 - OTIMIZADO
• Gerência de Configuração - CM • Medição e Análise - MA • Acompanhamento e Controle de Projeto - PMC • Planejamento de Projeto - PP • Garantia da Qualidade de Processo e Produto - PPQA • Gerenciamento de Requisitos - REQM • Gerenciamento de Acordo com Fornecedor - SAM	• Análise de Decisão e Resolução - DAR • Gerenciamento Integrado de Projeto - IPM • Definição de Processo Organizacional - OPD • Foco de Processo Organizacional - OPF • Integração de Produto - PI • Treinamento Organizacional - OT • Desenvolvimento de Requisitos - RD • Gerenciamento de Riscos - RSKM • Solução Técnica - TS • Verificação - VER • Validação - VAL	• Desempenho de Processo Organizacional - OPP • Gerenciamento Quantitativo de Projeto - QPM	• Gestão de Processo Organizacional - OPM • Análise Causal e Resolução - CAR

Saiba+
Para conhecer mais sobre o modelo CMMI, que é utilizado em organizações distribuídas por 94 países ao redor do mundo, basta acessar o site do CMMI Institute - www.cmmiinstitute.com

18

Apesar de mundialmente conhecido, o modelo CMMI não é fácil de ser implementado, sobretudo para organizações de pequeno e médio porte. Por este motivo, a SOFTEX – Associação para promoção do software brasileiro, com o intuito de impulsionar a melhoria no desenvolvimento de software das empresas brasileiras, decidiu criar o Modelo de Processo de Software Brasileiro, o MPS.BR.

A estrutura do MPS.BR-SW é similar a do CMMI-DEV, sendo dividida em estágios de maturidade, que vão do nível G ao nível A, em índice crescente. Assim como no modelo americano, existe uma série de pré-requisitos para avançar entre os estágios do MPS.BR. A tabela abaixo exibe os grupos de processos que são avaliados para avançar em cada um dos níveis. (Fonte Guia Geral de Software do Mps.Br).

NÍVEL DE MATURIDADE	PROCESSOS CHAVE
A - Em Otimização	• Não possui processos específicos, apenas implementação de atributos do processo.
B - Gerenciado Quantitativamente	• Gerência de Projetos – GPR (evolução)
C - Definido	• Gerência de Riscos – GRI • Desenvolvimento para Reutilização – DRU • Gerência de Decisões – GDE
D – Largamente Definido	• Verificação – VER • Validação – VAL • Projeto e Construção do Produto – PCP • Integração do Produto – ITP • Desenvolvimento de Requisitos – DRE
E – Parcialmente Definido	• Gerência de Projetos – GPR (evolução) • Gerência de Reutilização – GRU • Gerência de Recursos Humanos – GRH • Definição do Processo Organizacional – DFP • Avaliação e Melhoria do Processo Organizacional – AMP
F – Gerenciado	• Medição – MED • Garantia da Qualidade – GQA • Gerência de Portfólio de Projetos – GPP • Gerência de Configuração – GCO • Aquisição – AQU
G – Parcialmente Gerenciado	• Gerência de Requisitos – GRE • Gerência de Projetos – GPR

Como se pode observar, o modelo brasileiro contempla áreas similares ao modelo americano. Uma das grandes diferenças, entretanto, é o fato de que no MPS.BR há mais estágios de desenvolvimento, de modo que as exigências ficam melhor distribuídas entre os níveis, diminuindo a quantidade de requisitos necessários para avançar em cada uma das etapas.

Saiba+
Para conhecer mais sobre o Modelo de Processo de Software Brasileiro basta acessar o site da SOFTEX - http://www.softex.br/mpsbr/

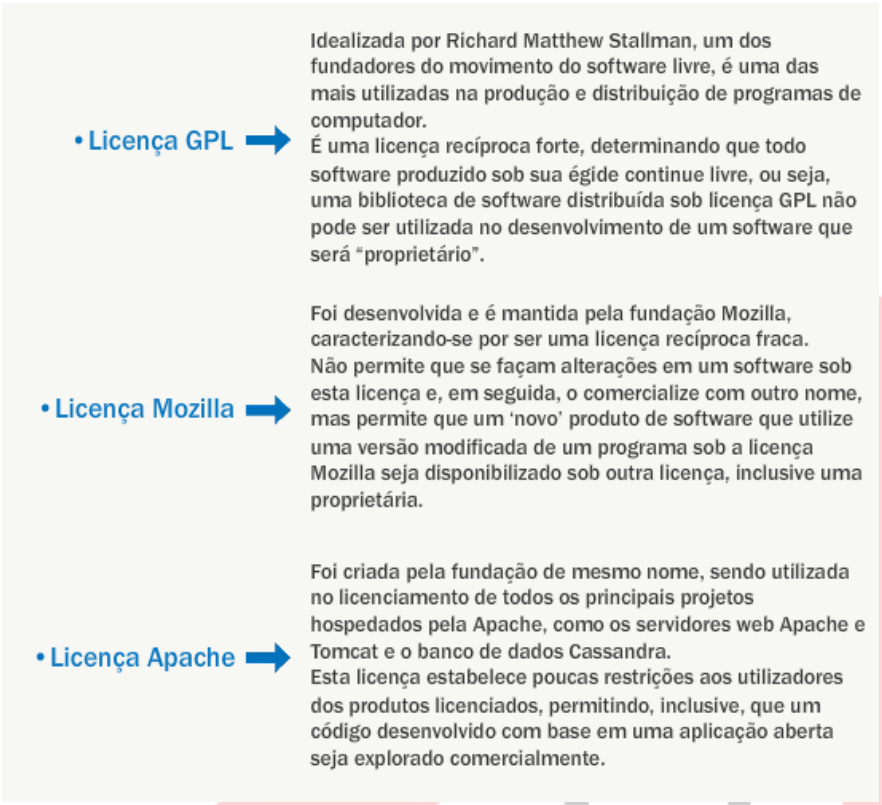
19

Licenças de Software

Normalmente, quando um software é construído, antes de se iniciar a sua distribuição, lhe é atribuída uma licença, que nada mais é do que a definição do que a pessoa que irá utilizar aquele software pode ou não fazer.

Existem duas principais categorias para as licenças, as que são associadas aos softwares proprietários, ou seja, que normalmente exigem pagamento para uso, e uma outra categoria, associada aos softwares gratuitos, ou livres, que tem como uma das características mais marcantes o fato de que os programas podem ser utilizados e distribuídos sem a necessidade de pagamento aos desenvolvedores.

Algumas das principais licenças de software livre são elencadas abaixo:



20

Resumo

O software, ou programa de computador, é uma sequência de instruções ou comandos lógicos que são executados ou interpretados por um computador com o intuito de executar uma tarefa específica. Apesar do primeiro programa de computador ter sido pensado no século XIX, apenas no início dos anos 1940 os softwares começaram a se apresentar como são hoje em dia. O avanço do hardware computacional foi um dos principais fatores para o avanço das linguagens de programação e, por conseguinte, do desenvolvimento de sistemas.

O avanço no uso dos dados na tecnologia da informação é visível nos dias atuais. A quantidade de informação gerada e manipulada por grandes expoentes do setor, como Google, Twitter e Facebook, só é possível graças aos consideráveis avanços não apenas nos equipamentos, mas nas estratégias de construção dos sistemas. O avanço do uso dos dados em TI veio concomitante ao desenvolvimento dos diversos paradigmas de desenvolvimento de software. Dentre os mais importantes, podemos citar o paradigma orientado a aspectos, o imperativo, o orientado a objetos, orientado a eventos e o lógico. Um paradigma de desenvolvimento de software fornece um exemplo geral, um modelo padrão para se pensar e se construir um programa de computador.

Um outro fato importante que marcou o mercado de software foi a criação dos modelos de maturidade do desenvolvimento de programas. Esta iniciativa representou uma guinada na forma como a qualidade era avaliada, deixando de dar ênfase apenas no produto para observar também a qualidade do processo.

O modelo de maturidade mais difundido é o Capability Maturity Model Integration, desenvolvido pela Universidade Carnegie Mellon e utilizado atualmente em mais de 90 países. O CMMI é subdividido em 5 níveis de maturidade do processo, compostos por 22 áreas chave. No Brasil, com o objetivo de tornar o processo de melhoria mais acessível as pequenas e médias empresas, o SOFTEX criou um modelo de maturidade nacional, denominado MPS.BR.

São dois os principais tipos de licença aplicados aos programas antes da sua distribuição. As licenças proprietárias são caracterizadas por exigir um pagamento do indivíduo ou empresa pelo uso do produto, além de que, frequentemente, o código fonte é inacessível para os clientes. Já as licenças livres, ou de código aberto, se caracterizam, prioritariamente, pelo livre uso dos programas sobre sua égide, que normalmente são distribuídos junto com o seu código fonte e sem custo.

UNIDADE 3 – CONTEXTUALIZAÇÃO DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

MÓDULO 2 – CRISE DE SOFTWERE

01

1 - CAUSAS E CONSEQUÊNCIAS DA CRISE DO SOFTWARE

Com o avanço da capacidade de processamento e armazenamento dos computadores e a criação das linguagens de programação de terceira geração, percebeu-se uma difusão em larga escala do uso dos computadores nos mais variados segmentos, muito por conta das novas possibilidades advindas com estas inovações. Na prática, notou-se que computadores e softwares, os quais até então eram utilizados como suporte a execução de tarefas secundárias, passaram a atuar nos principais processos de negócio da empresa, tornando-se uma vantagem competitiva das corporações.

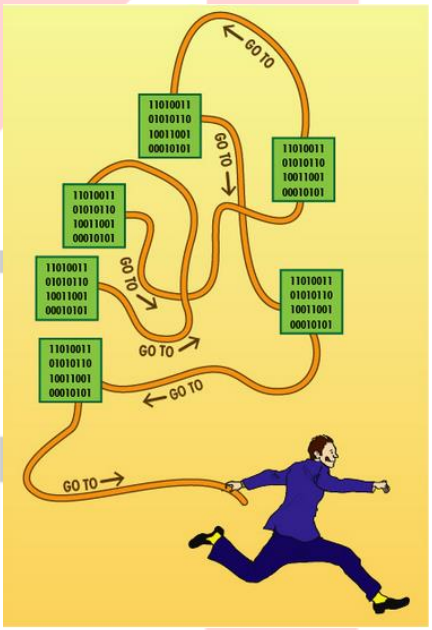
A pesquisa em linguagens de programação foi uma área que se desenvolveu muito a partir do final dos anos 50, se intensificando na década seguinte. Com o avanço das linguagens, as empresas passaram a desenvolver o seu próprio software, bibliotecas de código e até mesmo o sistema operacional dos seus computadores, sempre procurando adequar o processo de desenvolvimento às suas necessidades. Nesta época, percebe-se que o foco principal do processo de construção do software é o programador, profissional responsável por, sozinho, executar todas as etapas do desenvolvimento, desde a coleta das necessidades do usuário, passando pela codificação e execução de testes dos sistemas.

Foi justamente esta automatização de processos negociais importantes, associada à falta de maturidade no processo de desenvolvimento de programas, que gerou um dos mais graves problemas associados ao mercado de software já vistos, que ficou historicamente conhecido como a **Crise do Software**.

02

A primeira referência histórica que se tem do termo Crise do Software é originária de uma conferência de Engenharia de Software organizada pela OTAN, na Alemanha, no ano de 1968. Esta conferência reuniu programadores dos mais variados segmentos, oriundos de grandes corporações, da área governamental e até mesmo especialistas da academia, com o principal objetivo de estabelecer práticas mais maduras para o processo de desenvolvimento. Por essa razão o encontro é considerado por alguns autores como o nascimento da disciplina de Engenharia de Software.

03



Apesar de muitos dos participantes da conferência não se conhecerem, as discussões mostraram que os problemas que enfrentavam em seus projetos de desenvolvimento de software eram semelhantes e sempre descambavam pra estouro de prazo e custo, além de frequentemente apresentarem uma série de erros que comprometiam a utilização do produto final. Mas quais seriam, afinal, as causas destes problemas?

O que se percebeu, do meio para o final da década de 1960, foi que os programas começaram a crescer exponencialmente de tamanho e complexidade sem que houvesse um planejamento estruturado e sem que existisse um processo definido de avaliação da qualidade do processo.

A dependência dos programadores dificultava ainda mais o monitoramento da qualidade do desenvolvimento de sistemas. Não era incomum encontrar, em uma análise do código fonte das aplicações, o chamado **código espaguete** (*Spaghetti Code*).

O termo **código espaguete** surgiu a partir da observação de pedaços de código de programas que possuíam um fluxo extremamente confuso, onde as relações entre as diversas chamadas de funções e expressões eram tão emaranhadas que se assemelhavam a um prato de espaguete.

Em programas caracterizados por este tipo de código, é praticamente impossível adicionar ou modificar alguma coisa sem que, involuntariamente, seja alterado o funcionamento de outra funcionalidade, que não está relacionada à manutenção que está sendo executada.

04

A ocorrência do código espaguete está normalmente relacionada:

- à falta de supervisão no desenvolvimento de software,
- à inexistência da verificação da qualidade do código por amostragem e
- e à utilização de programadores inexperientes no desenvolvimento de sistemas complexos.

Outros três fatores fizeram os projetos de desenvolvimento de software constantemente ultrapassarem o orçamento definido para a sua execução, além de estourarem consideravelmente o tempo estimado de construção definido no cronograma. São eles:

- o aumento do tamanho e complexidade dos sistemas,
- a concentração do desenvolvimento apenas na mão dos programadores e
- o alto volume de erros nos sistemas.

Uma observação importante é que o problema não se limitava ao desenvolvimento de novos produtos de software. Sem uma **documentação completa** e sem um **código estruturado**, prover manutenção em sistemas legados era uma tarefa verdadeiramente hercúlea e, muitas vezes, dependente dos programadores que desenvolveram o código original.

05

Todos estes problemas causados pela crise do software iniciaram uma verdadeira revolução no modo com as empresas lidavam com a questão. A primeira quebra de paradigma foi a mudança de foco de estudo das ferramentas de desenvolvimento de código para o processo de construção de sistemas.

Esta iniciativa teve origem nas empresas e se expandiu em um curto espaço de tempo para as universidades. Estudos começaram a coletar e analisar dados dos projetos de software que não obtiveram sucesso, com o objetivo de identificar as principais causas do fracasso. Ao mesmo tempo, as organizações começaram a investir na criação de metodologias e diretrizes padrão para o seu processo de desenvolvimento de sistemas.

Um dos principais aprendizados oriundos desta pesquisa foi o de que a qualidade não pode ser considerada apenas na avaliação do produto final, mas durante todas as etapas do desenvolvimento. O ponto essencial de atenção passou a ser o **como o software é desenvolvido**.

Como resultado, técnicas como a **inspeção de código**, por exemplo, passaram a ser mais largamente utilizadas e demonstraram que, por vezes, eram mais efetivas do que a própria execução dos testes. Os modelos de ciclo de vida existentes passaram a ter mais uso e uma série de novos modelos foram propostos. A detecção de defeitos passou a ser realizada em estágios anteriores do processo de construção de software, ao tempo que as metodologias e técnicas de testes foram aperfeiçoadas.

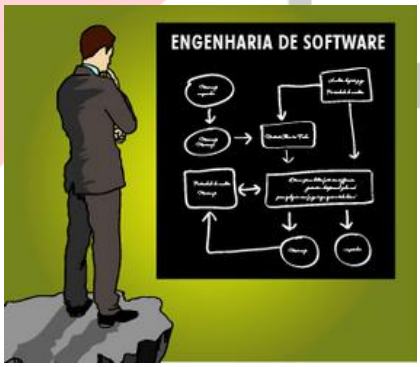
Os estudos destes novos métodos, práticas e técnicas foram, então, agrupados em uma única área de conhecimento, já existente, a **Engenharia de Software**.

06

2 - Engenharia de Software

No início, acreditava-se na tese de que a engenharia de software deveria ser pensada como a engenharia do hardware. Até então, o processo de construção de sistemas era extremamente baseado no esquema codificar-consertar, onde, sem se preocupar muito com o desenho da solução, os programadores iniciavam a codificação do sistema, trabalhando no esquema codificação de novas funcionalidades - conserto dos erros identificados.

A mudança de paradigma veio, como já apresentado, como uma das consequências da crise do software, que motivou uma série de estudos de suporte ao processo de construção dos sistemas. Este processo desvinculou o software do hardware, dando uma nova diretriz e um novo conceito ao que se tinha como engenharia de software.



Em um conceito atual, Engenharia de Software é a área do conhecimento que estuda métodos, ferramentas e técnicas estruturadas de suporte ao processo de desenvolvimento de software, com o intuito de tornar o processo mais racional, reduzindo os erros, produzindo software mais confiável e fazendo com que os projetos consigam manterem-se próximos das suas metas de prazo e custo.

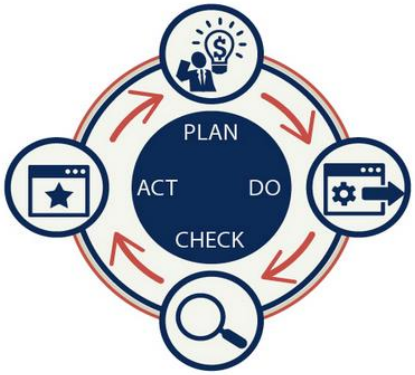
07

A engenharia de software, como já comentado, trouxe à tona o conceito de **qualidade do processo**, em detrimento apenas da qualidade do produto, mudando o conceito do que seria um programa de "boa qualidade".

Tomando como base apenas o produto final, poderíamos dizer que um bom software é aquele que implementa corretamente (sem erros) as funcionalidades solicitadas pelo cliente, dentro de um tempo de resposta aceitável, e sendo intuitivo para o usuário.

Será, entretanto, que a utilização de estruturas inadequadas de desenvolvimento, como o famoso código espaguete, não poderia ser considerado característica de um software ruim, mesmo que o produto final não apresente erros?

A qualidade do processo atua exatamente nesta área, monitorando e inspecionando o software durante toda a sua etapa de construção, de modo a avaliar se está sendo desenvolvido de acordo com os processos definidos e as boas práticas largamente difundidas. Quando a qualidade não é atingida por conta de um processo inadequado, pode-se, inclusive, desenvolver ações de melhoria contínua dos métodos e técnicas empregados, utilizando o modelo PDCA (*Plain* - planejar, *Do* - executar, *Check* - verificar, *Act* - agir) para fazer com que o processo atinja os índices de qualidade desejados.



Os métodos, técnicas e ferramentas, alvo de estudo da Engenharia de Software, são comumente organizados e estruturados em etapas, que são definidas como componentes de um processo maior, definido como "ciclo de vida do desenvolvimento de software". Alguns dos principais modelos de ciclo de vida serão apresentados nas seções subsequentes.

08

3 - Modelos de ciclo de vida de desenvolvimento de software

É sabido que não existe uma "bala de prata" para resolver qualquer problema de software, ou seja, uma metodologia, tecnologia ou técnica de gerenciamento única que consiga melhorar a produtividade do desenvolvimento e a confiabilidade de todos os produtos de software, independente da sua aplicação. Entretanto, podemos definir um portfólio de soluções, cujos elementos componentes seriam utilizados e adaptadas de acordo com o problema a ser enfrentado.

Esta observação nos leva à constatação de que não existe um ciclo de vida padrão para o desenvolvimento de sistemas, sendo que a escolha da melhor opção deve ser realizada a partir das características do projeto de software que será implementado.

São cinco os principais modelos de ciclo de vida de desenvolvimento de software:

- modelo em cascata,
- modelo de prototipação,
- modelo em espiral,
- modelo iterativo e incremental e
- modelo ágil.

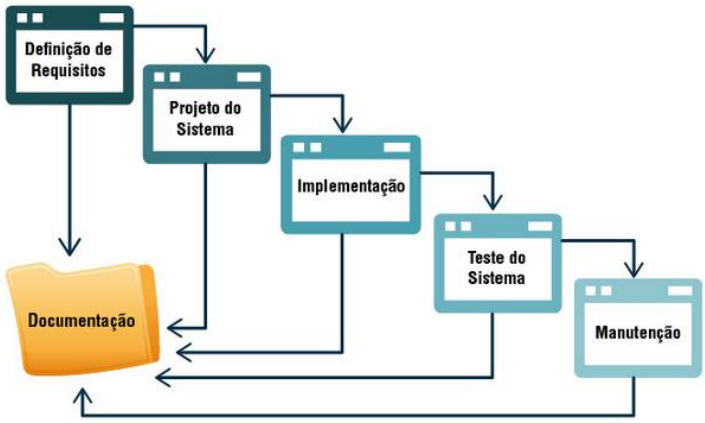
A seguir veremos em detalhes cada um desses modelos.

09

MODELO EM CASCATA

O modelo em cascata foi o primeiro processo de ciclo de vida formalmente estabelecido, sendo proposto no início da década de 70 com o objetivo de melhorar a forma de se gerenciar o desenvolvimento de grandes sistemas.

Tem como principal característica a execução sequencial das etapas de desenvolvimento, sendo que a evolução vai da atividade localizada no topo do processo até a que está localiza mais abaixo, assim como ocorre em uma cascata e que pode ser visualizado na imagem abaixo.



Exemplo de modelo de cascata

Por muito tempo, a construção de sistemas foi vista em torno de apenas duas etapas, a **análise do sistema** e a sua posterior **construção**. Em sua versão original, o propositor do modelo em cascata previu a existência de sete etapas sequenciais, mantendo as etapas de análise e codificação, mas acrescentando duas etapas de levantamento de requisitos, uma de desenho do programa, uma fase para testes e outra de operação do sistema. Neste modelo, uma fase só pode ser iniciada quando a anterior estiver totalmente completa.

Apesar de representar a primeira formalização de processos de construção de software, o modelo em cascata ainda hoje é alvo de muitas críticas. Uma das principais advém do fato de que ele não prevê a possibilidade de se voltar a fases anteriores durante o processo de desenvolvimento do sistema, o que, ao menos em tese, impede, por exemplo, uma gerência adequada das mudanças nos requisitos.

Modelo
O modelo em espiral foi proposto por Barry W. Boehm em 1988, no artigo “A spiral modelo f software develpoment and anhancement”, publicado pela ACM no congresso SIGSOFT Software Engineering Notes

10

MODELO DE PROTOTIPAÇÃO

Neste modelo, o desenvolvedor interage diretamente com o cliente durante todo o processo de desenvolvimento, levantando as necessidades e requisitos funcionais que o subsidiarão na construção e consequente aperfeiçoamento do protótipo, base para a construção do sistema que será posto em produção.

Prototipação é um método iterativo que gera, por vezes, uma redução no custo e no tempo necessário para desenvolvimento do sistema, trazendo o cliente para mais perto do processo de desenvolvimento, o que melhora a satisfação do usuário e facilita no entendimento dos requisitos, evitando retrabalho.

Por outro lado, nos casos em que os protótipos se tornam muito sofisticados, o que se percebe é um efeito contrário, sendo que o esforço desprendido nesta etapa é quase o mesmo do que seria aplicado na construção do sistema.

Outro ponto negativo é o fato de que, por visualizar o protótipo aparentemente funcional, os clientes esperam que a construção final do sistema seja breve, o que não acontece na prática, já que o protótipo é utilizado apenas como base para definição e validação das necessidades do cliente, sendo necessário se codificar do zero todo o sistema que entrará em produção.

11

MODELO EM ESPIRAL

Este modelo descreve o processo de construção de software em torno da interação de quatro fases de atividades, que podem ser personalizadas de modo a incorporar as mais novas técnicas e métodos do mercado.

Caracteriza-se por ser uma abordagem dirigida ao risco tendo surgido da experiência do refinamento da aplicação do modelo de ciclo de vida em cascata em uma série de projetos de desenvolvimento de sistemas.

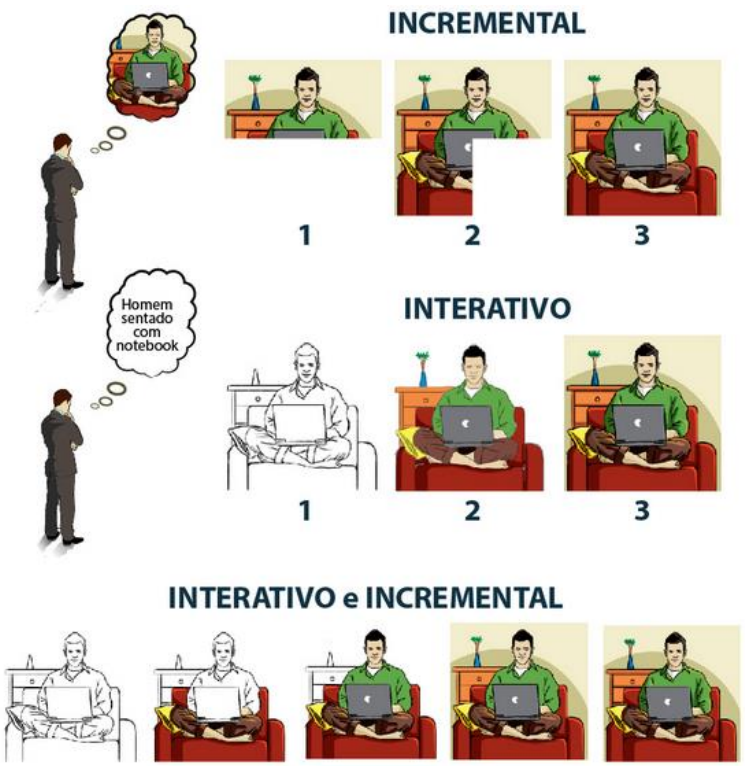
É importante ressaltar que este é um modelo incremental, ou seja, o software é dividido em partes que são entregues, uma de cada vez, ao final de cada uma das iterações.

A construção do software em espiral, criando iterações de desenvolvimento com foco na análise e controle dos riscos, auxilia na redução da ocorrência de erros de grande magnitude, já que existe um contínuo controle dos riscos e o tratamento dos erros é tratado em cada interação, não sendo necessário que todo o software seja entregue para que possa ser testado, como acontece no modelo em cascata.

12

MÉTODO ITERATIVO E INCREMENTAL

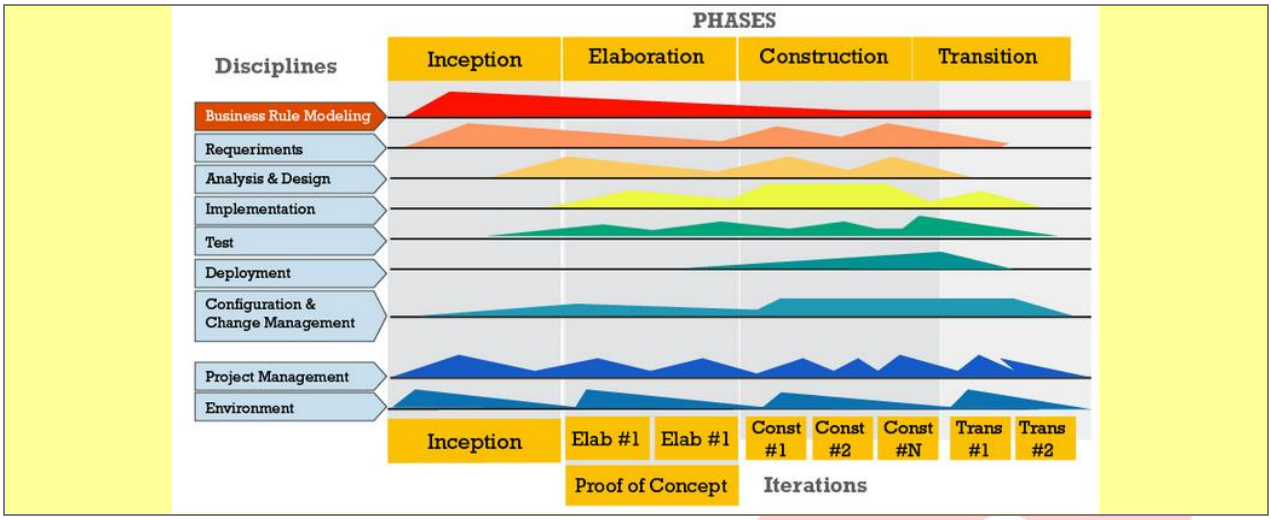
A base de funcionamento deste modelo é a divisão da construção do software em pedaços, que são desenvolvidos interativamente em etapas, sendo que durante mais de uma iteração pode estar acontecendo ao mesmo tempo durante o desenvolvimento, sendo integradas quando o trabalho de cada uma é finalizado.



Apesar de alguns defenderem que este método é recente e que se trata de uma evolução do modelo em espiral, descrições de projetos de software oriundos dos anos 70 e 80 demonstram que, ao menos na prática, há evidências de que elementos do processo de desenvolvimento iterativo e incremental estão presentes nos ciclos de desenvolvimento de software há algumas décadas.

O principal processo de software baseado no modelo Iterativo e Incremental é o RUP (IBM Rational Unified Process).

Saiba+
O RUP é uma abordagem proprietária desenvolvida pela IBM a partir da aquisição da Rational Software Corporation. O RUP é um framework completo de processo de software, trazendo modelos e práticas largamente utilizadas pelo mercado para construção e fornecimento de sistemas. É dividido em quatro fases de desenvolvimento, que são executadas de forma iterativa e incremental em torno de nove disciplinas de conhecimento, sendo seis de engenharia e três de suporte.



13

MODELO ÁGIL

O método de desenvolvimento ágil é baseado no modelo de desenvolvimento incremental. Apesar de alguns dos princípios do desenvolvimento ágil serem encontrados em iniciativas de diversas organizações desde o início dos anos 80, a formalização destes conceitos só aconteceu com a publicação do Manifesto Ágil, em fevereiro de 2001.

Neste ano, dezessete profissionais, especialistas nas mais variadas metodologias de programação, se reuniram com o objetivo de discutir sobre métodos de desenvolvimento mais leves, ou seja, com menos regras e mais fáceis de serem executados. O resultado deste encontro foi a escrita e divulgação do **Manifesto para o Desenvolvimento Ágil de Software**, que definiu quatro valores fundamentais na construção de software:

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

Desta forma, definem-se como aderente ao modelo ágil processos de ciclo de vida de desenvolvimento que estejam em conformidade com estes quatro valores.

14

4 - A CRISE DO SOFTWARE NOS TEMPOS MODERNOS

Mesmo com todo o avanço no estudo da engenharia de software, ainda é comum nos depararmos com erros na implementação de sistemas. Muitos pesquisadores acreditam que o uso de práticas e métodos largamente estabelecidos e testados realmente ajudaram na redução da ocorrência de “bugs” nos

programas, entretanto, defendem que acabar por completo com os erros de desenvolvimento é uma tarefa impossível, e que o sistema só se torna realmente estável com o seu uso constante e correção frequente dos erros.

Um “bug” do sistema ou do software é uma falha ou erro que pode levar o sistema a resultados inesperados e em desacordo com o comportamento desejado.

Este, inclusive, é um dos principais motivos pelos quais o setor bancário ainda mantém em funcionamento seus sistemas que foram desenvolvidos a várias décadas. Apesar de algumas vezes ainda utilizarem metodologias e técnicas consideradas hoje como arcaicas, os anos de utilização destes sistemas os fizeram extremamente seguros e confiáveis. A mesma lógica é encontrada nos softwares embarcados em aviões, que, após passar por anos de utilização, chegou a um estágio de estabilidade e confiabilidade, o que faz as empresas não quererem assumir o risco de introduzir novos componentes de software.

Esta preocupação com a ocorrência de erros é extremamente justificada. Imagine, por exemplo, o que poderia representar um erro no sistema bancário ou no software de controle de aviões. Os prejuízos financeiros poderiam ser incalculáveis, além de que milhares de vidas poderiam ser perdidas.
Para exemplificar o impacto que pode advir de um problema de software, listamos abaixo alguns dos mais conhecidos bugs da história.

15

Bug do Milênio

Foi um dos problemas de software que mais causou alvoroço ao redor do mundo. O grande medo era que, como muitos sistemas representavam os anos utilizando apenas os dois últimos caracteres, os computadores, com a virada do último milênio, acreditassem que tinham retornado ao ano de 1900, ao invés de avançado ao ano 2000, já que “00” representa os dois últimos dígitos de ambas as datas.

Isto poderia causar, por exemplo, erros na emissão de cobranças, aplicação de juros desproporcionais e inexistentes, panes no sistema de telefonia e de fornecimento de energia, além de outras consequências catastróficas inimagináveis.

Entretanto, o que se viu com a virada do milênio nem se aproximou do caos esperado. Os problemas decorrentes do bug foram pequenos e pontuais, não se sabe se por causa do esforço desprendido na correção do bug ou porque o problema era realmente menor do que se estimava.

16

Bug do Ano 2038

Quando todos pensavam que estavam livres das falhas de representação de datas nos computadores, como a que ocorreu no bug do milênio, surge o problema do ano 2038. Trata-se de uma falha na representação de datas em computadores, que pode causar erros em alguns programas de computador no ano de 2038.

O problema é encontrado apenas em sistemas que utilizam a representação de tempo POSIX, que utiliza uma estrutura de dados de 32-bits para armazenamento da data do sistema, e que é suficiente para armazenar 2147483647 segundos. Esta representação é padrão nos sistemas do tipo Unix e afeta a maioria dos sistemas, pois grande parte deste software foi desenvolvida na linguagem C.

Como a contagem do sistema se inicia em 1 de janeiro de 1970, temos como consequência que em 19 de janeiro de 2038 o contador do sistema atingirá o seu limite de 2147483647 segundos, o que pode causar uma série de problemas aos usuários, sobretudo por conta da imprecisão dos cálculos realizados pelo sistema.

17

Bug do Sistema de Defesa Patriot

Em fevereiro de 2001, uma bateria de mísseis de defesa Patriot operada na Arábia Saudita falhou no rastreamento e interceptação de um míssil SCUD disparado pelo Iraque. Como resultado do ataque, 28 soldados americanos morreram e aproximadamente 100 pessoas ficaram feridas.

Um relatório sobre o problema foi divulgado pela Divisão de gerenciamento da Informação e Tecnologia, vinculada ao Government Accountability Office – órgão responsável pelas investigações do congresso americano. Segundo o documento, a principal causa do problema foi uma falha no software no computador de controle de armas do sistema.

1. A precisão do sistema em detectar o deslocamento do o míssil SCUD é uma função da velocidade conhecida do míssil pelo tempo da última detecção do radar. No sistema Patriot, a velocidade é um número real representado em função de números inteiros e decimais, assim como o tempo é continuamente mantido pelo relógio interno do sistema e representado em décimos de segundos. O tempo era contado em função do momento em que o sistema era reiniciado, de modo que, quanto maior o tempo que permanecesse ligado, maior o tamanho do número de representação do tempo.

O que aconteceu foi que, como o sistema utilizava apenas 24 bits para armazenamento do tempo, e como a bateria de defesa já estava em funcionamento há um tempo considerável, o espaço de memória disponível não foi suficiente para armazenar corretamente tempo, que perdeu precisão, causando falha no cálculo necessário para acionamento do sistema.

18

BLACKOUT NORTE AMERICANO DE 2003

O mais sério blackout ocorrido nos Estados unidos, e o segundo maior da história recente, foi decorrente da combinação de erro humano com falhas dos equipamentos.

Por conta da alta demanda por energia em uma determinada linha de alta voltagem, localizada no norte de Ohio, os cabos de transmissão esquentaram e se expandiram, fazendo com que aumentassem de peso e, consequentemente, caíssem sobre árvores presentes no solo, levando ao desligamento do sistema.

Durante a próxima meia hora, outras três linhas, pelo mesmo motivo, também esbarraram em árvores e se desligaram, sobrecarregando o sistema como um todo, que acabou desligado pois, sem estas fontes de energia, já não conseguia dar vazão a demanda por energia.

A questão principal é que, normalmente, este tipo de problema dispara um alarme na sala de controle do sistema de energia, de modo a direcionar a atuação da equipe de manutenção e suporte. Entretanto, **por uma falha no sistema de alarmes**, mesmo com a condição crítica, nenhum sinal sonoro ou visual foi emitido pelo alarme do sistema, que simplesmente travou silenciosamente, parando de processar os alertas e deixando os operadores desavisados da causa do problema.

A falha do alarme fez com que mais de 55 milhões de pessoas nos EUA e Canadá ficassem sem energia por até dois dias, gerando um prejuízo estimado em mais de seis bilhões de dólares.

19

RESUMO

A crise do software foi um fenômeno que se originou do aumento do tamanho e complexidade dos sistemas, sem que houvesse um planejamento estruturado e sem que existisse um processo definido de avaliação da qualidade do processo de construção de sistemas.

Este processo teve como consequência direta um aumento dos estudos na área de Engenharia de Software, sobretudo com o objetivo de incrementar os métodos, processos e tecnologias utilizadas no desenvolvimento de sistemas.

Os métodos, técnicas e ferramentas, alvos de estudo da Engenharia de Software, são comumente organizados e estruturados em etapas, que são definidas como componentes de um processo maior, definido como “ciclo de vida do desenvolvimento de software”. O principal modelo de ciclo de vida formalmente definido foi o modelo em cascata. Outros modelos importantes são: o modelo de prototipação, o modelo em espiral, o iterativo e incremental e o modelo ágil.

Mesmo com todo o avanço observado no estudo da Engenharia de Software, ainda é comum nos depararmos com erros na implementação de sistemas, já que, apesar da existência de uma série de padrões e modelos, trata-se de uma tarefa que é extremamente dependente do componente humano. Alguns dos principais problemas de software da história são o bug do milênio, o problema do ano 2038, o bug do sistema de defesa patriot e o blackout americano de 2003.

UNIDADE 3 – CONTEXTUALIZAÇÃO DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

MÓDULO 3 – BENEFÍCIOS DA UTILIZAÇÃO SISTEMAS DE INFORMAÇÃO

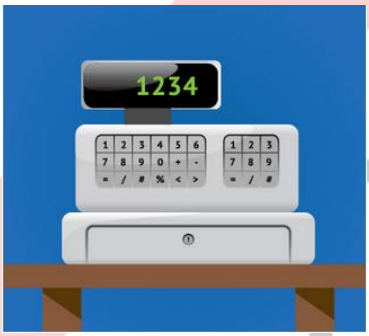
01

1 - TECNOLOGIA DA INFORMAÇÃO NOS BANCOS

A chegada dos computadores e programas representou uma das maiores quebras de paradigma da idade moderna. A informatização de processos e métodos iniciou uma nova era, tornando as empresas e governos mais eficientes e eficazes.

Atualmente, não se pode ver a automatização dos processos como uma vantagem competitiva, mas como um meio de sobrevivência no mercado. É difícil que uma empresa que ainda trabalhe da forma convencional consiga competir com concorrentes que estão no auge da automatização. A queda no preço dos equipamentos associada ao aumento no número de profissionais só tem incentivado cada vez mais a ampliação e profissionalização do setor de TI nas empresas.

Você consegue imaginar um grande supermercado em que os caixas ainda utilizem calculadoras e caderninhos para registrar as suas compras e fazer o balanço de pagamentos? Ou um banco em que você só consiga fazer operações na sua agência de origem? Será que ainda conseguimos imaginar como é que se faz a declaração do imposto de renda naqueles formulários antigos? E como seria viver hoje sem as benesses da educação a distância? Nas próximas seções vamos analisar um pouco do impacto da TI em cada uma destas importantes áreas.



02

Caixas eletrônicos, cartões de crédito e débito, poder fazer transações em qualquer lugar, essa são só algumas das facilidades trazidas pela introdução da Tecnologia da Informação do setor bancário. Mas o que hoje parece tão trivial, é uma realidade que só foi introduzida nas últimas décadas. No Brasil, por exemplo, a chegada dos caixas eletrônicos só aconteceu na década de 1980.

Hoje os bancos formam uma grande rede, não só entre agências de um mesmo banco, mas entre diferentes instituições financeiras. Os centros de processamentos de dados foram aperfeiçoados e se tornaram os mais seguros dentre as diversas categorias de empresas. Hoje é praticamente impossível chegar a uma cidade de pequeno a médio porte e não encontrar sequer um correspondente bancário onde se possa efetuar um saque ou pagar uma conta, tudo on-line e em tempo real.

Já são mais de 100 milhões de contas correntes no Brasil, o equivalente a quase 50% de nossa população. Além disso, das mais de 389 bilhões de transações realizadas, 23% foram executadas apenas através dos caixas eletrônicos. Isso demonstra o alcance do setor e o impacto da informatização nas transações bancárias.

Saiba+
Leia mais sobre a evolução de contas correntes no Brasil em http://www.valor.com.br/financas/3531404/numero-de-contas-correntes-no-brasil-cresce-6-em-2013-103-milhoes

03

Além de trazer facilidade para os usuários, a automatização bancária gerou produtividade e eficiência para os bancos. As operações começaram a ser executadas de forma mais rápida e com menor dependência do fator humano. Automatizar os serviços deixou de ser uma vantagem competitiva e se tornou um fator primordial para que a instituição conseguisse manter-se no mercado.

O lado negativo deste processo é a redução dos empregos no setor bancário, fenômeno já historicamente relacionado a informatização de processos e atividades. Os números comprovam que, mesmo apresentando um elevado crescimento no número de contas, quantidade de cartões emitidos e transações realizadas, a quantidade de funcionários praticamente se mantém a mesma.

Saiba+
Assista à reportagem publicada na década de 1980, que exibe a surpresa do brasileiro ao se deparar com as inovações no setor bancário. http://www.youtube.com/watch?v=LtgIXD8bpJc

04

INTERNET BANKING

Uma das principais variantes da automatização bancária é o chamado “internet banking”. Esta novidade fez com que os usuários pudessem executar uma série de transações a qualquer hora do dia e sem precisar sequer ir a uma agência bancária.

Hoje em dia, é quase impossível pensar em uma instituição bancária que não disponibilize seus serviços através da internet, fato que tem levado os bancos ao mundo web e feito deste setor o maior investidor em tecnologia do mercado brasileiro, com gastos na ordem de 20 bilhões de reais.

O uso do internet banking tem apresentado um crescimento exponencial nos últimos anos. Segundo dados da Febraban, 47% de todas as transações bancárias de 2013 foram realizadas pelos internet e mobile banking, conquanto apenas 37% utilizaram os canais "ditos" como tradicionais, a exemplo dos já citados caixas eletrônicos e agências.

São diversos os benefícios oriundos deste novo modelo de atendimento bancário. Se a satisfação do usuário é aumentada pelo fato de que passa a ter acesso aos serviços 24 horas por dia, os custos bancários são reduzidos com a diminuição da necessidade de agências e da contratação de pessoal.

Saiba+
Os dados sobre o internet banking são expressivos, sobretudo porque em 2009 a participação deste canal sobre o total de transações era de apenas 31%, ou seja, em apenas quatro anos percebeu-se um crescimento de 16 pontos percentuais nas estatísticas de utilização deste serviço. Foram quase 42 milhões de contas correntes acessadas via web em 2013, e mais de 16,6 bilhões de transações realizadas por este canal apenas neste ano.
Leia mais em: http://www.febraban.org.br/Noticias1.asp?id_texto=2364&id_pagina=86&palavra=

05

2 - Tecnologia da Informação no Comércio Varejista

A automatização trouxe enormes benefícios para o setor varejista, sobretudo no monitoramento e controle das operações. Operações de balanço de vendas e controle de estoque, que antes exigiam alto número de pessoas para execução, hoje estão disponíveis em apenas um clique. O controle de preços dos produtos, cálculo da lucratividade das empresas e dos tributos, controle dos funcionários e lojas, tudo hoje é feito de forma automatizada nas grandes redes.

E como imaginar uma forma outra forma de gerenciar milhares de lojas e funcionários, espalhadas por todo o Brasil? Os números do setor realmente impressionam, o maior grupo do Brasil, o Pão de Açúcar, possui cerca de 1882 lojas e 150 mil funcionários. Já o Walmart possui mais de 540 lojas e 82 mil funcionários e a Ricardo Eletro mais de 1100 lojas e 30 mil funcionários. Juntas, as 100 maiores redes de varejo faturaram mais de 300 bilhões de reais em 2012.

A automatização dos seus processos internos também trouxe eficiência e eficácia às operações das grandes redes. Uma das mudanças mais visíveis foi o aumento da velocidade no atendimento aos clientes com a utilização de modernos leitores de códigos de barras para registos dos produtos no sistema.

Em um passado não muito distante, cabia aos caixas digitarem o preço de cada um dos produtos, o que, além de tornar o processo mais moroso, tradicionalmente originava erros de digitação.

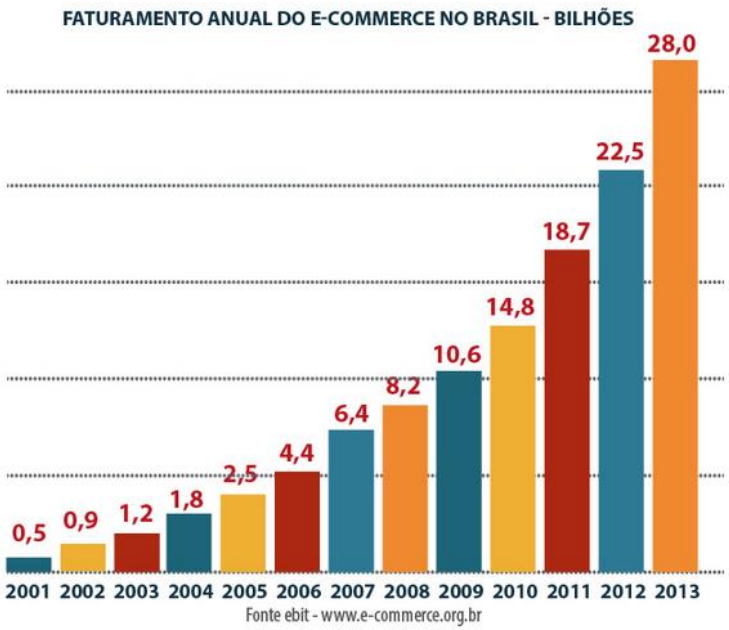
Saiba+
Para saber mais, leia
http://exame.abril.com.br/negocios/noticias/25-maiores-varejistas-no-brasil-segundo-ibevar#8

06

A principal revolução observada nos últimos anos, entretanto, diz respeito à nova onda chamada de **comércio eletrônico**. Por muito tempo existiu um mito de que operações de vendas on-line nunca seriam bem sucedidas, pois o consumidor sempre teve por hábito escolher o produto pessoalmente, observando as suas características e tratando diretamente com os vendedores. Além disso, como as pessoas iriam acreditar que um determinado produto comprado em um site da internet realmente era verdadeiro? Como saber se realmente vai ser entregue em sua residência ou se está apenas sendo vítima de um golpe de estelionato?

Entretanto, todos esses medos que permeavam o início das operações ficaram apenas nas suposições, muito porque a geração conectada na rede mundial tem um perfil diferente do que se imaginava.

Ao contrário do que se previu, o *e-commerce* despontou como uma verdadeira febre entre os consumidores. As facilidades da compra on-line, que pode ser realizada a qualquer hora e sem precisar sair de casa, turbinaram as vendas das empresas, fazendo o faturamento em comércio eletrônico, que era de 0,5 bilhão em 2001, passasse a alcançar 28 bilhões de reais em 2013, um aumento de 5.500% em pouco mais de 10 anos.



Fonte: http://www.e-commerce.org.br/stats.php

07

Este grande avanço acabou trazendo as tradicionais redes de comércio varejista para o mundo virtual, como o Walmart, Ricardo Eletro e Casas Bahia, o que tem acirrando ainda mais a disputa e vem incrementando os números do setor. Dados recentes apontam que apenas no primeiro semestre de 2014 já houve um crescimento de 26% das vendas on-line em comparação com igual período do ano passado. Já em relação ao varejo convencional, o IBGE apontou crescimento de apenas 5% no mesmo período.

08

SISTEMAS DE PAGAMENTOS

Não é só na atividade finalística que o comércio eletrônico tem provocado avanços. Toda a cadeia de pagamentos tem procurado encontrar métodos mais seguros para a execução das vendas on-line. É neste nicho de mercado que surgiram o UOL Pag Seguro e o PayPal.

Estas ferramentas agem como um intermediário confiável na transação, permitindo que as operações sejam realizadas sem que o vendedor tenha acesso aos seus dados de pagamento como, por exemplo, o número do seu cartão de crédito. Esta é uma ótima alternativa para compras que são realizadas em lojas virtuais pequenas, onde não há uma garantia da boa fé do vendedor, ou em compras em lojas internacionais.

UOL Pag Seguro
Para conhecer melhor estas ferramentas, acesse https://pagseguro.uol.com.br/

PayPal
Para conhecer melhor esta ferramenta, acesse www.paypal.com

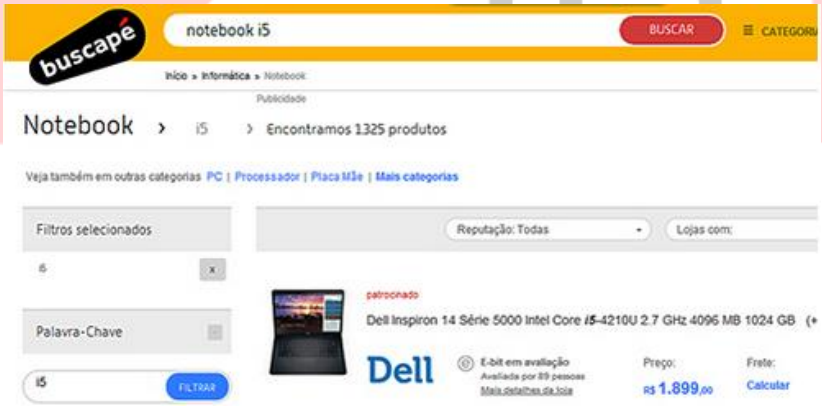
09

Sistemas de Cotação de Preços

Outra inovação impulsionada pelo avanço do comércio eletrônico foram os sistemas de cotação de preços on-line. Através da utilização desta ferramenta, é possível, em um único lugar, buscar o preço de um produto em diversas lojas virtuais, auxiliando o comprador a descobrir quem vende mais barato.

Os sistemas fornecem, ainda, uma série de opções de filtro da busca, como a seleção de uma determinada marca ou uma característica específica do produto, além de permitir ordenar a busca pela reputação da loja ou pela avaliação do produto que está sendo comercializado.

Dentre os principais sites brasileiros, pode-se citar o buscape.com.br, o zoom.com.br e o Bondfaro.com.br



10

Sistemas de Recomendação

Os sistemas de recomendação são uma das maiores novidades do comércio eletrônico. Na verdade, não se caracterizam por serem sistemas autônomos, mas por serem melhorias nas lojas virtuais.

A principal intenção dos sistemas de recomendação é tentar entender as necessidades do usuário, fazendo com que a sua experiência seja a melhor possível e que tenha fácil acesso aos produtos que está procurando. Para executar esta estratégia, diversas abordagens são adotadas, sendo as principais:

• observar o **histórico de compras dos usuários**,
• as **relações entre diferentes produtos** ou
• realizar uma **análise semântica da experiência individual** de cada usuário na internet.

Na primeira abordagem, o sistema observa qual o padrão de compra dos usuários que realizaram uma compra no site. Observe, na imagem abaixo, que, ao acessar a página para compra de um determinado produto, é oferecido ao usuário informações sobre outros produtos que foram comprados por usuários que acessaram esta mesma página.

Este processo de recomendação atualmente é considerado como trivial, sendo relativamente fácil de ser implementado. Todos os dados de transações se encontram em um banco de dados que organiza os dados de forma estruturada, o que facilita o acesso das informações e o consequente processamento das recomendações.

Dados Estruturados e Não Estruturados

Dados Estruturados – são dados normalmente organizados em tabelas de bancos de dados. Uma tabela de banco de dados se assemelha a uma planilha eletrônica, onde existe uma linha de rótulos, que identifica o tipo dos dados, e uma série de outras linhas, que representam os dados propriamente ditos. Observando a tabela abaixo, a máquina pode claramente diferenciar os dados que representam os nomes, dos que indicam data de nascimento e naturalidade.

NOME	DATA DE NASCIMENTO	NATURALIDADE
Carlos Nascimento Silva	02/01/1976	Salvador
Joana D'arc de Almeida	02/09/1988	São Paulo
Tiago Angelo Arruda	29/01/1970	Recife
Maria da Silva Carvalho	19/08/2972	Porto Alegre

Dados Não Estruturados – são dados que não estão organizados em tabelas ou qualquer outra estrutura que permita o seu processamento automático de forma direta. Observe, por exemplo, o texto abaixo. Ao computador, diferente do entendimento humano, não é trivial descobrir qual o significado da data "02/01/1976", da cidade "Salvador" ou até mesmo do nome "Caio Nascimento da Silva", já que a informação se encontra de forma "não estruturada", ou seja, dispersa em um parágrafo de texto livre.

> O jornalista Carlos Nascimento da Silva, nascido em 02/01/1976 na cidade de Salvador, publicou o seu último livro, o Romance Nas asas da Paixão, na última quinta-feira.

11

Outra forma comumente utilizada na recomendação é a **definição manual** da relação entre diferentes produtos. Imagine, por exemplo, que, ao comprar uma televisão, o sistema alertasse que talvez você também precise comprar um suporte de parede. O sistema podia, ainda, ir além e lhe oferecer a compra de um *home theater* de última geração ou até mesmo de um aparelho Blue-ray com desconto no caso da compra dos dois produtos. O que se percebe é que esta abordagem auxilia na venda de produtos que muitas vezes são necessários, mas acabariam esquecidos pelos usuários.

12

A última estratégia de recomendação automática é a análise semântica da experiência individual do usuário.

Trazendo da definição do dicionário para a computação, dizemos que a análise semântica de uma palavra permite a definição do seu significado a partir do contexto em que ela se encontra, ou seja, da sua relação com as outras palavras presentes na frase.

Entender a semântica é algo que é trivial para nós, humanos. Entretanto, as máquinas têm dificuldade em entender o contexto das coisas. Na verdade, os computadores têm dificuldade em entender palavras ambíguas, cujo sentido depende do contexto em que aparece.

Por exemplo, segundo o dicionário Michaelis da língua Portuguesa, a palavra **manga**, dentre outras definições, pode ser:

1. Parte do vestuário que cobre o braço, cingindo-o.
2. Fruta da mangueira.

Para saber de qual definição de manga estamos tratando, é necessário, portanto, que definamos o contexto de utilização da palavra. Desta forma, teríamos:

1. Tenho que cortar a **manga** desta camisa. *(Parte do vestuário que cobre o braço, cingindo-o.)*
2. Acho que esta **manga** está madura! *(Fruta da mangueira)*

É justamente neste ponto que os computadores tradicionais "travam". Diferentemente de como acontece com os homens, as máquinas não conseguem inferir qual o contexto em que a palavra está inserida. Foi esta deficiência que motivou uma série de pesquisas na área de inteligência artificial, voltadas a fazer com que os computadores possam entender o contexto das frases.

13

Algumas das alternativas de se descobrir as necessidades do usuário são conhecidas e, muitas vezes, são controversas. Um dos exemplos é do Gmail, serviço de correio eletrônico do Google, que faz a varredura dos e-mails do usuário, identificando os tópicos de interesse de modo a direcionar as suas propagandas. Observando a figura abaixo, observamos que, ao acessar um e-mail que trata de uma conferência em gerenciamento de projetos, são oferecidos ao usuário anúncios correlacionados com o tópico "projetos", como a venda de uma ferramenta de gestão de projetos.

14

Outra forma é aplicar a análise semântica das redes sociais, de modo a entender o sentido das postagens do usuário e as conexões entre pessoas, tópicos e produtos. Utilizando a inteligência na análise destas redes sociais, é possível construir uma grande base de dados semântica, que permite a execução de recomendações personalizadas aos usuários, sobre produtos ou qualquer outro tópico de interesse.

É com base nesta análise das interações dos usuários nas redes sociais que foi desenvolvida a nova ferramenta de busca do Walmart – o Polaris. O novo buscador se utiliza justamente da tecnologia semântica da base de dados, que utiliza sinais identificados da mineração das redes sociais para ordenar os resultados da busca. O algoritmo leva em consideração, por exemplo, a quantidade de “curtidas” que um produto do Walmart teve no Facebook ou se o usuário ou alguém do seu círculo de amigos fez algum comentário sobre o produto.

Só com a implantação deste novo buscador, houve, segundo o Walmart, um aumento de 10 a 15% na probabilidade de que o consumidor completasse a sua compra na loja virtual da empresa, números estes bastante expressivos.

15

3 - Tecnologia da Informação no Governo

Acompanhando a tendência observada na iniciativa privada, governos por todo o mundo passaram a informatizar seus processos e métodos. As antigas máquinas de escrever, cujo som era tão característico nas repartições públicas, acabou tornando-se artigo raro, sendo substituídas pelos computadores.

Além da informatização física das repartições públicas, outra inovação advinda da tecnologia foi o surgimento do **governo eletrônico**, ou *e-gov*.

O conceito de *e-gov* busca levar os serviços públicos além da fronteira dos órgãos, utilizando a internet como principal ferramenta para propiciar ao cidadão uma maior aproximação com o governo.

Nesta linha, objetivos como a migração dos serviços públicos para o ambiente *on-line*, a publicação de dados dos órgãos do governo em portais públicos e o estímulo à utilização de serviços de governo eletrônicos por parte dos cidadãos têm sido tratados como diretrizes da agenda de governo eletrônico em muitos países, dentre eles o Brasil.

16

No Brasil, o Portal de Serviços eletrônicos do governo federal – www.servicos.gov.br, foi lançado no ano de 2012 com o objetivo de se tornar o guia eletrônico de serviços governamentais.

O portal já possui mais de 597 serviços cadastrados, referentes as categorias G2C, G2E, G2B e G2G, sendo estes providos por mais de 20 órgãos do governo federal.

Através do portal, de nossa casa ou outro local, temos acesso a uma infinidade de serviços públicos.

Além dos benefícios advindos com o maior acesso dos cidadãos aos serviços públicos, a implantação do governo eletrônico também tem-se caraterizado por permitir a redução de gastos públicos.

Saiba+
Os serviços de Governo eletrônico se dividem em quatro tipos, a depender do ator que está interagindo com o governo:
• **Governo-para-Cidadão (G2C)** - tem por objetivo facilitar o processo de comunicação entre o governo e os cidadãos;
• **Governo-para-Empresas (G2B)** - visa facilitar a interação com as empresas, eliminando a redundância na coleta de dados e reduzindo os custos para o governo;
• **Governo-para-Funcionários (G2E)** – tem relação com os serviços prestados pelo governo a funcionários ou empregados públicos;
• **Governo-para-Governo (G2G)** – voltado ao processo de cooperação e colaboração entre governos de diferentes níveis.

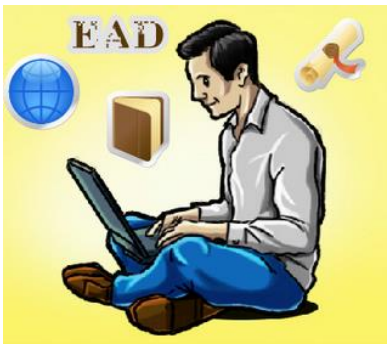
Serviços públicos
Alguns serviços disponíveis:
• Fazer a inscrição no Cadastro Nacional de Pessoa Física, da Receita Federal.
• Acompanhar o andamento dos processos de concessão inicial de benefícios junto ao INSS.
• Cadastrar ocorrência de roubo de veículos no Sistema Alerta da Polícia Rodoviária Federal.
• Agendar a emissão do passaporte junto a Polícia Federal.
• Solicitar a emissão de uma série de certidões, como a de contribuições previdenciárias e de nada consta na Receita Federal.

17

4 - Tecnologia da Informação na Educação - Educação a Distância

O que se percebe é que a educação a distância vem crescendo em ritmo mais avançado do que a educação presencial. Segundo dados do censo da Educação Superior divulgados pelo Ministério da Educação, de 2011 para 2012 verificou-se um incremento de 12,2% nas matrículas EAD, conquanto a educação presencial teve um aumento de apenas 3,1%.

Para se ter ideia do fenômeno da EAD no Brasil, o número de alunos de graduação na modalidade passou de pouco mais de 1500 alunos em 2000 para mais de 5,7 milhões de alunos em 2012, um crescimento de quase 380.000% em apenas 12 anos.



Esses números só comprovam o impacto da implantação desta nova ferramenta educacional, que tem contribuído para expansão do ensino superior, fazendo-o chegar aos maiores rincões do nosso país. Hoje, não é preciso sair das pequenas cidades e se deslocar aos grandes centros para se conseguir terminar um bom curso. Além disso, com a flexibilidade no acesso às aulas, o curso pode ser realizado a qualquer hora, sem que isto atrapalhe o dia a dia do trabalho, bastando apenas um computador e acesso à internet.

Esta realidade tem permitido a um grande número de jovens, sobretudo os com menos recursos, continuar os seus estudos, aumentando o nível da mão de obra local e elevando a educação no Brasil.

18

RESUMO

A informatização de processos e métodos iniciou uma nova era, tornando empresas e governos mais eficientes e eficazes. A automatização trouxe uma série de facilidades para os consumidores e usuários, ao tempo que causou uma considerável redução de custos por parte das empresas.

O impacto da Tecnologia da Informação em diversos setores foi marcante, gerando uma verdadeira revolução nos modos de operação. Exemplo importante são as mudanças no setor bancário, com a chegada dos caixas eletrônicos, a implantação dos cartões de crédito, além da chegada do *internet banking*. Outro exemplo importante é a automatização das operações de varejo e a chegada do comércio eletrônico, que fez que o meio virtual se tornasse um dos lugares mais propícios para a compra de produtos.

E não foi apenas no mundo privado que a implantação da TI trouxe benefícios, o setor público também foi afetado positivamente, sendo o governo eletrônico o grande exemplo. O e-gov foi fundamental na aproximação entre governo e cidadão, facilitando a interação entre estes dois atores e auxiliando na redução dos gastos públicos.

Por fim, temos a Educação a Distância como uma das principais consequências da TI no setor educacional. A criação dos cursos de EAD permitiu a difusão do ensino universitário por cidades pequenas e médias, permitindo uma maior capacitação da população e melhorando os índices de desempenho social do nosso país.

UNIDADE 3 – CONTEXTUALIZAÇÃO DO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

MÓDULO 4 – PRINCIPAIS TECNOLOGIAS UTILIZADAS PARA GERAR SISTEMAS DE INFORMAÇÃO

01

1 - MODELAGEM DE NEGÓCIO

O processo de construção do software se inicia com o entendimento das necessidades do cliente. Nesta etapa, uma equipe fica responsável por identificar e documentar estes dados, que são chamados de requisitos do sistema, que são a base para a modelagem do negócio.

A modelagem de negócio é um processo de alto nível que tem por objetivo desenhar como a solução de tecnologia da informação será desenvolvida. Imagine a modelagem do negócio como uma atividade de entendimento do problema a ser enfrentado e a sua posterior representação em elementos de sistema.

São duas as principais abordagens para modelagem e desenho de sistemas,

- a metodologia orientada a objeto (OO), que costuma utilizar a linguagem de notação UML,
- e a metodologia orientada a serviço (SOA), que costuma utilizar o modelo BPM e a notação BPMN.

Veremos a seguir cada uma dessas abordagens.

Processo de alto nível
Associando com o conceito das linguagens de alto nível, um processo de alto nível é aquele cuja representação é mais próxima do entendimento humano do que da máquina.

UML
A Unified Modeling Language é uma linguagem de modelagem de Sistemas Orientados a Objeto. Foi criada a partir da junção das três principais linguagens de modelagem utilizadas à época – o método Booch, o OMT e o OOSE.

BPM
Business Process Modeling, ou modelagem de processos de negócio, é a atividade de identificar e documentar (desenhar) os processos de negócio de uma empresa, com o objetivo de padronizar os procedimentos e realizar a melhoria constante dos processos.

BPMN
Business Process Modeling Notation é a principal notação utilizada no desenho dos processos de acordo com o BPM.

02

Na abordagem **Orientada a Objetos**, os requisitos são representados em função de uma série de diagramas, sendo os principais:

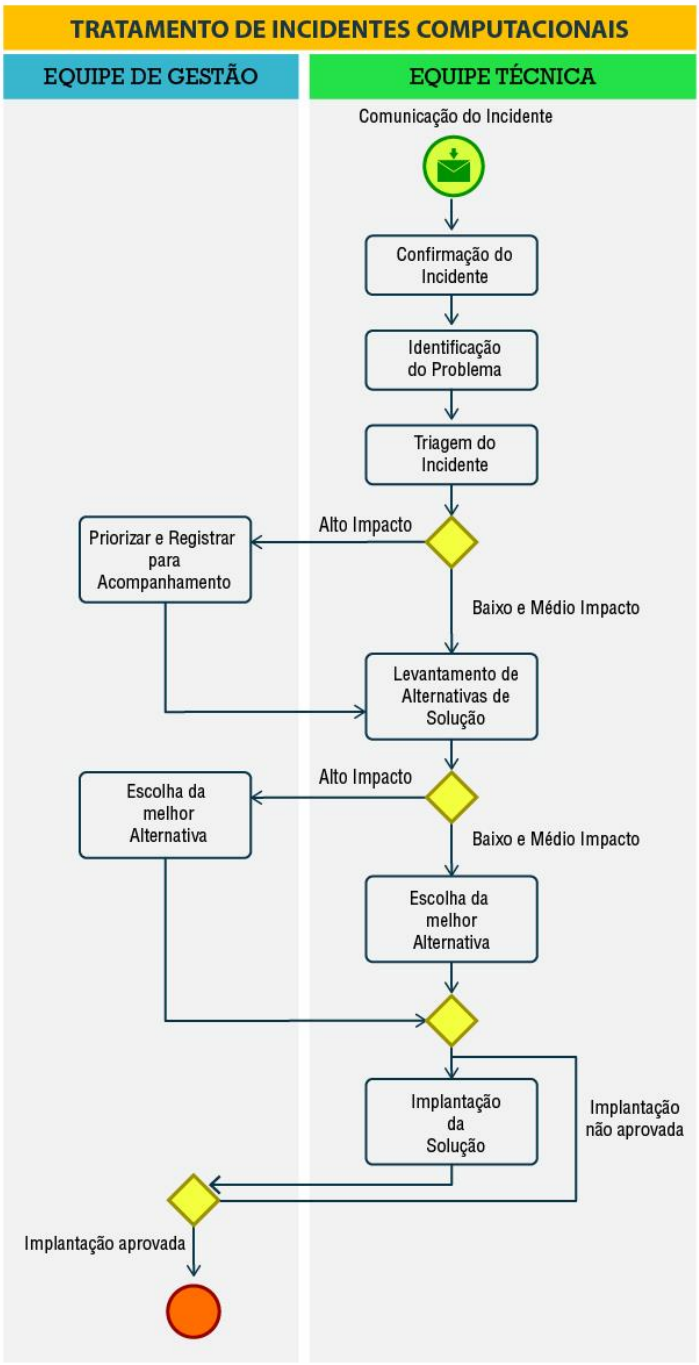
- **Diagrama de casos de uso** → Representa os atores e suas interações com os casos de uso do sistema. Os casos de uso representam um grupo de ações executadas em fluxos com intuito de atingir um determinado objetivo.
- **Diagrama de classes** → Utilizada para representar as classes do sistema e as suas relações. Uma classe é uma representação de um elemento do mundo real.
- **Diagrama de sequência** → Representa uma sequência ordenada de ações, por vezes de diferentes classes, executadas com o objetivo de fornecer uma visão do passo a passo necessário para realizar uma determinada ação.
- **Diagrama de atividade** → É um diagrama de controle que auxilia no entendimento do fluxo do sistema que será desenvolvido.

03

A outra abordagem a ser discutida é a **Orientação a Serviços**, que busca representar o sistema em torno dos serviços que serão providos aos clientes.

Este processo de modelagem se inicia com a identificação da cadeia de valor pela equipe de analistas, ou seja, são levantados os processos de mais alto nível dentre aqueles que irão compor o escopo do sistema. A partir daí é realizado um refinamento, onde são identificadas e documentadas (ou mapeadas) as interações entre diferentes atores na execução de atividades que compõem cada processo.

Como já comentado, a notação mais utilizada para documentar os processos de negócio é o Business Process Model and Notation (BPMN). Na figura abaixo, apresentamos o modelo em BPMN do mapeamento de um processo de exemplo.

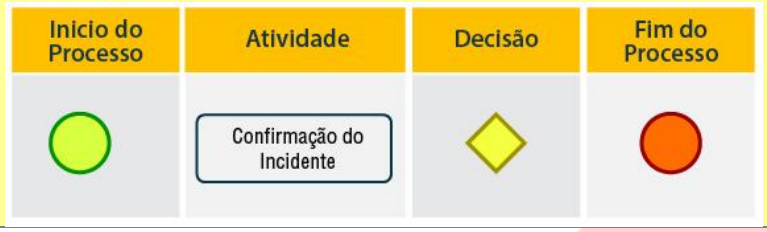


Observe que o processo mapeado é composto por uma sequência de atividades, desempenhadas por dois atores, a “Equipe de Gestão” e a “Equipe Técnica”. Cada ator é representado em uma estrutura separada, chamada de “raia”.

O processo se inicia com a comunicação do acidente e termina com a aprovação da implantação da solução. Durante a sua execução, percebe-se que o fluxo é definido por estruturas de decisão, a partir da análise dos valores de entrada. Por exemplo, se um incidente é classificado como de “baixo ou médio

impacto", o fluxo segue para a atividade "Levantamento de Alternativas de Solução", caso contrário, segue para a atividade "Priorizar e Registrar para Acompanhamento". Saiba+

Resumo da Notação BPMN



Cadeia de valor

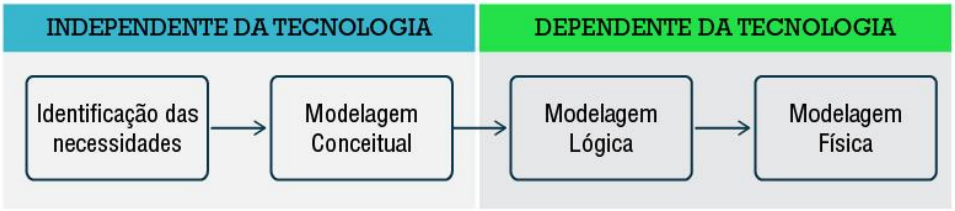
É o conjunto de atividades executadas por uma organização e que são necessárias para que um determinado produto ou serviço seja entregue.

Saiba+
A partir do desenho do processo são identificados os serviços candidatos, ou seja, os que serão efetivamente implementados. Para cada serviço é estabelecido um contrato de serviço, que define, dentre outros elementos, sua forma de funcionamento e nível de qualidade esperado. Esta é a principal especificação utilizada para a modelagem dos dados orientada a serviço e a posterior codificação do sistema.

04

2 - Modelagem de Dados

Após a modelagem do negócio, a próxima etapa a ser realizada para a construção do sistema é a modelagem dos dados. Este processo, de desenho da estrutura de armazenamento das informações no sistema, é separado em atividades sequenciais, que vão desde a identificação das necessidades do sistema a partir dos modelos de negócio até a modelagem física do banco de dados. Estas quatro etapas podem ser associadas duas a duas, criando um grupo de atividades que é dependente da tecnologia que será utilizada no banco de dados e outro grupo que é independente da tecnologia.



05

Identificação das Necessidades

O processo de identificação das necessidades é voltado ao esclarecimento das estruturas de dados necessárias para construção do sistema. Para isso, costuma-se utilizar a documentação e diagramas produzidos durante a etapa de modelagem de negócio, bem como os requisitos levantados durante o levantamento das necessidades do cliente.

- **Modelo conceitual**

É a primeira etapa do desenho do modelo de dados, devendo utilizar uma linguagem que seja de fácil entendimento por parte do cliente. A ideia é se preocupar apenas com as características e relacionamentos dos objetos, que devem tentar se aproximar ao máximo dos conceitos do mundo real, sem se deixar influenciar pelas limitações impostas pelas arquiteturas técnicas dos bancos de dados. Desta forma, o foco da atenção é com o que está sendo representado, e não como será implementado tecnicamente. Como já dito, é fundamental o entendimento de que a estruturação do modelo conceitual independe da tecnologia de banco de dados que será utilizada.

Uma das principais abstrações utilizadas na representação do modelo conceitual é o **Diagrama de Entidade Relacionamento** (DER). Este diagrama tem por princípios:

- Documentar as regras de negócio de forma a facilitar o entendimento dos clientes e usuários.
- É construído com base em elementos do mundo real – as entidades.
- Representa de forma gráfica os relacionamentos entre os elementos.

06

Uma **entidade** representa um conceito do mundo real, seja um evento ou um objeto, sobre o qual devem ser mantidas informações ou registros. Imagine, por exemplo, que na modelagem de um sistema de uma biblioteca, teremos uma entidade chamada “Livro”, que conterá os registos dos livros fisicamente encontrados.

REPRESENTAÇÃO GRÁFICA

Um **atributo** representa uma característica associada a uma entidade. Desta forma, a nossa entidade “Livro” teria atributos como título, autor, editora e ano de publicação.

REPRESENTAÇÃO GRÁFICA

Um **relacionamento** representa uma ligação existente entre diferentes entidades. Poderíamos dizer, por exemplo, que um livro, por exemplo, que um livro "possui" um exemplar.

REPRESENTAÇÃO GRÁFICA

Cardinalidade é uma característica que define o nível dos relacionamentos entre as entidades. São três os principais tipos:

- Um-para-muitos (1:N);
- Muitos-para-muitos (N:N);
- Um-para-um (1:1).

Um-para-muitos (1:N)

Um-para-muitos (1:N) – "uma" entidade A está relacionada a "N" entidades B.

Muitos-para-muitos (N:N)

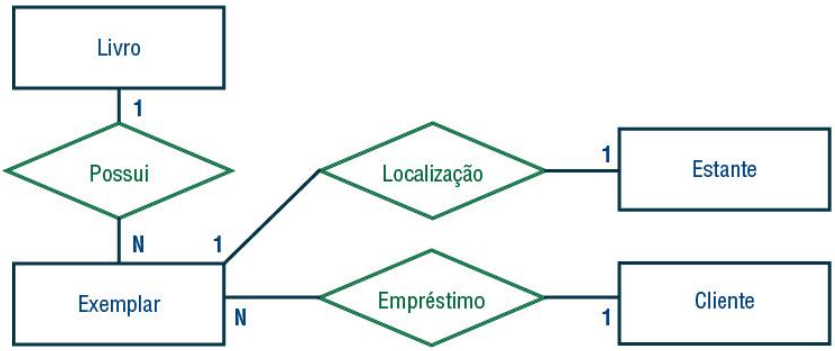
Muitos-para-muitos (N:N) – "uma" entidade A está associada a "N" entidades B e "uma" entidade B está associada a "N" entidades A.

Um-para-um (1:1)

Um-para-um (1:1) – "uma" entidade A está relacionada a "uma" entidade B.

07

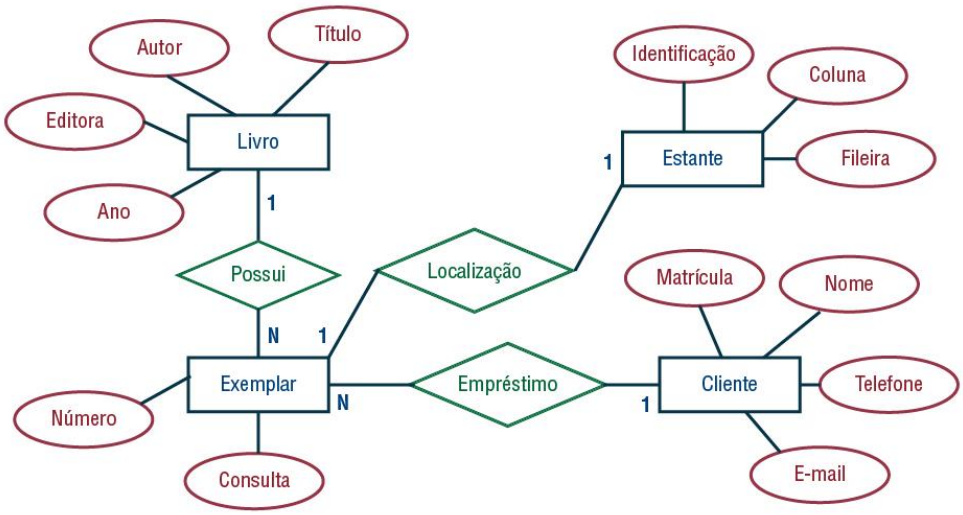
Continuemos, então, a modelagem conceitual do nosso sistema de bibliotecas. Além da entidade "Livro", que possui "n" "Exemplares", temos a informação de que cada exemplar só pode estar em apenas "uma" "Estante", e que "Clientes" podem realizar o empréstimo de "n" exemplares ao mesmo tempo. Desta forma, utilizando a notação proposta, teríamos o seguinte Diagrama de Entidade Relacionamento.



Perceba que nenhuma destas definições faz qualquer referência a arquitetura técnica do banco de dados onde os dados serão armazenados ou sobre a tecnologia que será utilizada, mas aponta apenas objetos e seus relacionamentos.

08

A próxima etapa da modelagem conceitual é a definição dos atributos das entidades já elencadas no modelo. Já sabemos que um Livro tem título, autor, editora e ano de publicação. Uma Estante, por sua vez, tem um código de identificação, o número da coluna e o número da fileira em que está localizada. O exemplar tem um número do exemplar e a informação se é apenas de consulta. Por fim, o cliente tem como atributos matrícula, nome, telefone e e-mail. Aplicando estas informações ao nosso diagrama, passaríamos a ter a seguinte representação:



Com a definição das entidades, seus relacionamentos e cardinalidade, além dos atributos, chegamos ao fim do desenho conceitual dos dados.

09

- **Modelo Lógico**

O modelo lógico é um modelo intermediário entre a representação conceitual e o modelo físico. Aqui, a representação das entidades já passa a incorporar características ligadas a algum tipo de tecnologia, com a utilização de chaves primárias e integridade referencial. É uma visão mais próxima dos profissionais que serão responsáveis por implementar o banco de dados do que do cliente responsável pelo fornecimento dos requisitos.

O processo de construção do modelo lógico tem como base o produto gerado na modelagem conceitual. Desta forma, as atividades desta etapa buscam derivar os relacionamentos do modelo conceitual, o aprofundando em detalhes, e a execução da normalização dos dados. Como regras de derivação, temos, por exemplo:

- Em Relacionamentos “um-para-um”, a chave primária da tabela mais importante (Exemplar) se torna chave estrangeira da outra (Estante).

- Em relacionamentos “um-para-muitos”, a chave primária da tabela Cliente se torna chave estrangeira da tabela Exemplar.

- No caso de relacionamentos muitos-para-muitos, deve-se criar uma terceira tabela que irá associar as chaves primárias das duas tabelas originais.

Uma **Chave primária** pode ser composta por apenas um ou por um conjunto de elementos associados. Tem por objetivo garantir que o valor dos seus componentes associados será sempre único para cada conjunto de dados, o que garante a unicidade deste registro.
Imagine um cadastro de veículos onde consta o modelo do veículo, o ano e a Cor. Estas três características sempre se repetem, de modo que seria impossível identificar unicamente um determinado veículo dentro do grupo de registros. Desta forma, é comum utilizar neste cadastro a informação da placa do veículo como chave primária, já que existe a garantia de que as placas nunca se repetem, o que garante que este registro será único dentro do conjunto de dados.

Normalização

É um conjunto de regras que é aplicado ao modelo de dados de forma a evitar erros, como a duplicação de dados e a mistura de diferentes assuntos em uma mesma entidade. Cada regra da normalização é chamada de “Forma Normal”. As três principais Formas Normais são apresentadas abaixo.

Primeira Forma Normal (1FN) - todos os atributos da entidade devem ser atômicos. Um campo de uma tabela que armazena mais de um valor, por exemplo, deve ser transformado em outra tabela.

Observe a tabela Filiais apresentada abaixo. Será que está de acordo com a 1FN?

Tabela Filiais

CNPJ	Nome Fantasia	Endereços das filiais
123456789234	Super Empresa ltda.	Av. Silva Lima, n 123, Arapongas Av São João n 478, São Sebastião R. Faria Lima, n 15, São Paulo

A Tabela não está de acordo com a 1FN, já que existe mais de um endereço dentro de um mesmo registro, o que quebra a regra da atomicidade. Desta forma, para normalizar a tabela filiais, precisaríamos segmentar o campo "Endereço das Filiais", criando uma segunda tabela só para armazenar endereços. A normalização da tabela Filiais de acordo com a Primeira Forma Normal é apresentada abaixo.

Tabela Filiais

CNPJ	Nome Fantasia	Endereços das filiais
123456789234	Super Empresa ltda.	Av. Silva Lima, n 123, Arapongas Av São João n 478, São Sebastião R. Faria Lima, n 15, São Paulo

Tabela Endereços

Id Endereço	CNPJ	Endereços das filiais
1	123456789234	Av. Silva Lima, n 123, Arapongas
2	123456789234	Av São João n 478, São Sebastião
3	123456789234	R. Faria Lima, n 15, São Paulo

Segunda Forma Normal (2FN) – para estar na 2FN, a tabela deve ser em concordância com a 1FN e todos os seus atributos devem ser relacionados a todos os campos que compõe a sua chave primária (que é composta). Tomemos como exemplos a Tabela Exemplar.

Tabela Exemplar

Id Exemplar (Chave Primária)	Número do Exemplar	Id livro (Chave Primária)	Título do Livro	Autor	Ano
87981278	1	817298	A hora da Estrela	Clarice Lispector	1977
87981279	2	817298	A hora da Estrela	Clarice Lispector	1977
89801278	3	817298	A hora da Estrela	Clarice Lispector	1977
11278078	1	129809	Capitães de Areia	Jorge Amado	1937

Observe que temos uma chave primária composta pelo Id Exemplar e pelo Id Livro. Entretanto, alguns atributos estão relacionados a apenas uma parte da chave primária. O título, o autor e o ano, por exemplo, estão relacionados apenas ao campo Id Livro, não tendo relação com o Id Exemplar, campo que também faz parte da chave primária. Desta forma, para normalizar esta tabela devemos segmentá-la em duas tabelas, separando os interesses em conjuntos diversos, conforme exibido abaixo:

Tabela Exemplar

Id Exemplar (Chave	Número do	Consulta	Código do

Primária)	Exemplar		Livro (Chave estrangeira)
87981278	1	Sim	817298
87981279	2	Não	817298
89801278	3	Não	817298
11278078	1	Sim	129809

Tabela Livro

Código do Livro (Chave Primária)	Título do Livro	Autor	Ano
817298	A hora da Estrela	Clarice Lispector	1977
129809	Capitães de Areia	Jorge Amado	1937

Terceira Forma Normal (3FN) – para estar na 3FN, a tabela deve estar em concordância com a 2FN e todos os seus atributos que não são chaves primárias devem ser dependentes da chave primária estabelecida. Por exemplo, imagine que tivéssemos a seguinte tabela:

Tabela Exemplar

Id Exemplar (Chave Primária)	Número do Exemplar	Consulta	Título do Livro	Autor	Ano
87981278	1	Sim	A hora da Estrela	Clarice Lispector	1977
87981279	2	Não	A hora da Estrela	Clarice Lispector	1977
89801278	3	Não	A hora da Estrela	Clarice Lispector	1977
11278078	1	Sim	Capitães de Areia	Jorge Amado	1937

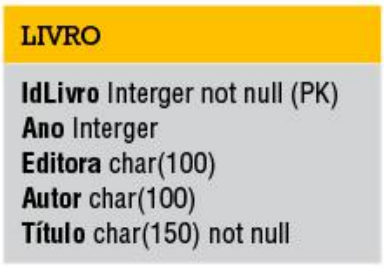
Observe que é possível identificar que há uma repetição desnecessária dos dados do livro. Isso ocorre porque os campos título do livro, autor e ano não estão diretamente relacionados a chave primária da tabela. Desta forma, para tornar o exemplo de acordo com a terceira forma normal, seria necessário segmentar a tabela em duas – uma para exemplar e outra para livro, e criar um relacionamento entre elas, nos moldes do que foi apresentado no exercício da segunda forma normal.

10

- **Modelo Físico**

Ligado diretamente à implementação do banco de dados. Tem um nível de abstração mais baixo, sendo pouco intuitivo para o usuário. Este é o momento de detalhar a organização dos dados de acordo com a estrutura física do Sistema Gerenciador de Banco de Dados, como tipos dos campos, índices etc.

Um exemplo da representação física da tabela livro é exibida abaixo. Observe que ao lado do nome de cada campo há uma referência do tipo de dados que irá armazenar, como o “integer”, que armazena números, e o “char”, que armazena caracteres de texto.



11

3 - LINGUAGENS DE PROGRAMAÇÃO

Após as etapas de modelagem do negócio e dos dados, se inicia o processo de codificação do sistema propriamente dito. São centenas as linguagens de programação atualmente existentes no mundo, atendendo aos mais variados propósitos e implementando diversos dos paradigmas já apresentados. Desta forma, antes de mais nada, é necessário definir qual linguagem de programação será utilizada para codificar o sistema.

Algumas das principais linguagens utilizadas atualmente são descritas a seguir.

Linguagem C

A linguagem C é uma linguagem de terceira geração criada na Bell Labs, no ano de 1972. É a linguagem de programação mais utilizada no mundo, sendo a base para a construção de sistemas muito conhecidos, como os sistemas operacionais Windows e Linux.

É uma linguagem ligada ao paradigma imperativo, que, como já apresentado, tem como peças chave a existência de variáveis, de operações de atribuição, a execução sequencial dos comandos e a forma iterativa de repetição.

Um exemplo do código fonte de um programa em linguagem C é exibido abaixo. O resultado da execução deste código é a impressão do texto “Alô Mundo” na tela.

```
#include <stdio.h>
int main()
{

printf(“Alô Mundo”);

}
```

Em uma análise rápida do código, temos que o comando #include permite, ao programa em execução, o uso de funcionalidades existentes na biblioteca stdio.h, ao tempo que o comando printf executa a operação de imprimir o texto na tela. Toda linha de execução de comandos em C é finalizada com a utilização do ponto e vírgula.

12

Java

O Java está entre as três linguagens de programação mais utilizadas atualmente em todo o mundo. Foi criada originalmente pela Sun, empresa que foi recentemente adquirida pela Oracle. Em 2007 todo o código de suporte à linguagem foi publicado como software livre através da licença GPL.

Assim como a "C", é um exemplo de linguagem de terceira geração, só que, ao invés do paradigma imperativo, é vinculada ao paradigma de orientação a objetos, ou seja, trabalha com a ideia da representação de elementos do mundo real. Para isso, utiliza estruturas conhecidas como Classes, que são compostas por atributos e operações.

Quando de sua concepção, a linguagem Java foi pensada para funcionar nos mais diversos dispositivos, dos mais potentes servidores até equipamentos com menos capacidade de processamento, como dispositivos móveis, alguns eletrodomésticos inteligentes ou aplicações embarcadas em chips. Esta realidade é possível, sobretudo, pelo fato de que é uma linguagem portável, ou seja, o código gerado pode ser executado em diversos dispositivos por ser esta uma linguagem interpretada, ao invés de compilada.

Uma linguagem **interpretada** se caracteriza por não ser executada diretamente pelo Sistema Operacional, mas através de um aplicativo intermediário denominado interpretador. Desta forma, uma vez este interpretador instalado, um único código produzido pode ser executado em diversas plataformas, sem a necessidade de uma programação específica ou a realização de adaptações de código.

C#

O C# (ou C Sharp) é, assim como o Java, uma linguagem de programação orientada a objetos. Foi criada em 2000 como parte do framework .NET da Microsoft.

Apesar de derivada da Linguagem C, possui características importantes, como a gerência de alocação de memória volátil, onde objetos armazenados que não são mais utilizados pelo programa em execução são automaticamente removidos, liberando espaço em memória para armazenamento de outros elementos do sistema.

Licença GPL
É uma das licenças mais utilizadas na produção e distribuição de programas de computador. É uma licença recíproca forte, determinando que todo software produzido sob sua égide continue livre, ou seja, uma biblioteca de software distribuída sob licença GPL não pode ser utilizada no desenvolvimento de um software que será "proprietário".

Saiba+
O principal ambiente de programação utilizado para se codificar programas em C# é a IDE Visual Studio. A versão Express desta ferramenta de desenvolvimento integrado é disponibilizada gratuitamente pela Microsoft no site http://www.visualstudio.com/

13

PhP

É uma linguagem que foi originalmente pensada para a Web, com o objetivo de gerar conteúdo dinâmico nas páginas html. Por conta disso, alguns a consideram uma linguagem de domínio específico, ou de quarta geração.

Com o passar dos anos, foi adaptada para ser utilizada também na construção de aplicações Desktop, extensão que ficou conhecida como PHP-GTK. Entretanto, apesar de ter se mostrado estável, ainda não conseguiu se posicionar dentre as linguagens líderes nesta área, assim como fez na programação Web.

O Php é executado sobre uma arquitetura gratuita e de código aberto. Tradicionalmente, o código php é embutido dentro do código html da página web através da utilização das tags “<?php” e “?>”. Abaixo podemos observar um exemplo de uma página html com código em php embutido. A ação deste código é escrever na tela do navegador o texto “Alô Mundo!”.

```
<html>
  <head>
    <title>Teste do Php</title>
  </head>
   <body>
    <?php Print "Alô Mundo!"; ?>
   </body>
</html>
```

O Php se destaca pela sua velocidade e simplicidade de codificação, sendo que o seu desempenho é diretamente dependente do servidor de aplicação utilizado para executar a aplicação desenvolvida.

Aplicações desktop
As **aplicações desktop** são as aplicações tradicionais, que são instaladas e executadas diretamente nos computadores de trabalho. Já as **aplicações web** são aplicações instaladas e executadas a partir de um servidor web, sendo que o acesso do usuário ao sistema normalmente é realizado através do uso de um navegador web, como o Firefox, Internet Explorer ou Chrome.

14

4 - FRAMEWORK

Um *framework* é uma estrutura que tem por objetivo congregar características, funções comuns, que são encontradas em diversos aplicativos e que, por conseguinte, podem ser reaproveitadas. Alguns dos principais **benefícios** do uso dos *frameworks* são que:

- Tornam mais fácil o trabalho com tarefas que são consideradas complexas, já que não há necessidade de desenvolver novamente o código, que é reaproveitado.
- Os frameworks mais difundidos possuem código estável e com menor ocorrência de falhas.
- Proporcionam uma redução no tempo e no esforço necessário para a construção dos novos programas.
- Facilitam o entendimento do sistema em função da proliferação do uso das funções em diferentes aplicações de diferentes empresas.
- O código já está desenvolvido, já foi testado exaustivamente e melhorado.

Por outro lado, algumas **desvantagens** também são observadas:

- Em função de ser construído de forma genérica, para atender uma ampla variedade de cenários, muitas vezes o código não oferece a melhor performance.
- O uso do framework depende de uma curva de aprendizado por parte dos programadores da empresa
- Por serem utilizados em uma grande quantidade de aplicações, em caso de ocorrência de erros, estes terão um impacto maior nos sistemas da empresa.

Existem **frameworks** com os mais diferentes objetivos, como o de dar suporte a persistência de dados na aplicação, a exibição de interfaces web, implementação de controles de segurança e autenticação, geração de relatórios e gráficos, dentre outros. Observe que estas funcionalidades, como já exposto, são comumente encontradas nas mais diversas aplicações, independente do problema que elas tentam resolver.

15

Resumo

O processo de construção de software é dividido em etapas, que vão do levantamento das necessidades do usuário até a codificação do sistema. Nesta linha, os modelos avançam de uma representação mais próxima da linguagem do cliente, o que facilita o entendimento do negócio, até um modelo mais voltado ao programador, mais técnico e voltado para implementação.

A primeira etapa é a modelagem do negócio, que se baseia fortemente nos requisitos levantados na atividade de elicitação e análise. É na modelagem do negócio que ocorre o desenho do sistema, são representadas as suas funcionalidades e correlação com as necessidades do cliente.

Atualmente são duas as principais metodologias para modelagem do negócio, a Orientada a Objetos e a Orientada a Serviços. A maioria dos projetos orientados a objetos são construídos com base na Unified Modeling Language, ou UML. Esta representação possui uma série de diagramas que auxiliam na

identificação dos objetos do sistema, dos seus atributos, métodos e inter-relações. Dentre os diagramas mais utilizados estão o Diagrama de Casos de Uso, o Diagrama de Classes, o Diagrama de Sequência e o Diagrama de Atividades.

A outra abordagem apresentada foi a Orientada a Serviços, que tem o seu processo baseado na identificação e mapeamento dos processos de negócio. A notação mais utilizada para mapeamento e detalhamento destes processos é a *Business Process Model and Notation* ou BPMN.

É a partir destes processos que são identificados os serviços candidatos, ou seja, os serviços que serão codificados e entregues ao cliente. Para cada um destes serviços candidatos é especificado um contrato de serviço, que se apresenta como o principal documento de especificação e é base para o entendimento do programador sobre o que deve ser implementado.

Após a etapa de modelagem do negócio, temos a modelagem de dados do sistema, que é dividida em três atividades, a modelagem conceitual, lógica e física. A modelagem conceitual dos dados é voltada para esclarecimento das necessidades do cliente, a conceitual já apresenta algum aprofundamento técnico, e a física é voltada a dar suporte ao processo de implantação do esquema em um banco de dados.

Após todo o processo de modelagem, do negócio e dos dados, inicia-se o processo de codificação do sistema. Para tanto, uma atividade essencial é definir qual a linguagem de programação que será utilizada. Dentre as mais conhecidas, destaca-se a linguagem “C”, Java, C# e Php.

É interessante ressaltar que, apesar de divididas em fases, muitas das etapas podem ocorrer parcialmente em paralelo, de forma iterativa e incremental. Isto vai depender do paradigma de desenvolvimento selecionado.