

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/341327460>

Function Point Analysis (FPA)

Article · January 2017

CITATIONS

0

READS

76

1 author:



[Alvaro F. Pinheiro](#)

University of Pernambuco

29 PUBLICATIONS 5 CITATIONS

SEE PROFILE

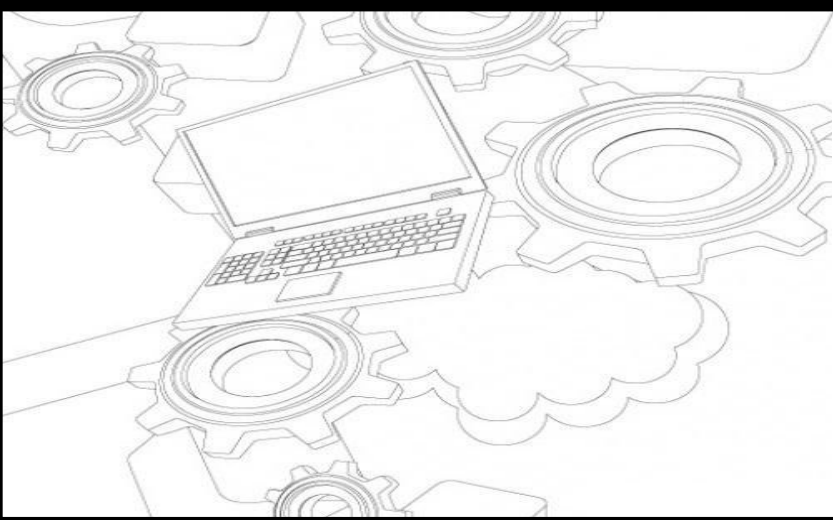
Álvaro Farias Pinheiro

Revista Fundamentos da Engenharia de Software

Análise de Ponto de Função

APF

Ano II – Número 3
Março de 2017



Série Fundamentos da Engenharia de Software

Análise de Ponto de Função

PINHEIRO, Álvaro Farias
Autor

Série Fundamentos da Engenharia de Software

Análise de Ponto de Função

Publicação 2017

O autor acredita que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, por ventura, mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação. Eventuais erratas estarão disponíveis para download no site de publicação.

As imagens utilizadas neste livro foram obtidas na Internet.

Dados da Publicação

Pinheiro, Álvaro Farias

Série Fundamentos da Engenharia de Software: Análise de Ponto de Função

Ano II – Número 3 – Recife, Março de 2017

Selo Editorial: Publicação Independente

1. APF Introdução
2. APF Conceitos
3. APF Terminologias
4. APF Métrica

Série Fundamentos da Engenharia de Software

Análise de Ponto de Função

Publicação Independente
Revista em português com o título
Análise de Ponto de Função
Série Fundamentos da Engenharia de Software
Ano II – Número 3
Recife – Pernambuco – Brasil
Março de 2017

Introdução

Uma dúvida que me perseguiu por muito tempo foi, como determinar o prazo e custo do desenvolvimento de um produto que ainda não temos todos os requisitos levantados e sim a previsão do que deverá ser feito. A resposta foi chegando aos poucos, com as tentativas frustradas de estimar prazos e custos de algo que na sua totalidade era desconhecido. Assim, fui percebendo que não há como determinar o tamanho de um *software*, por mais que se possua experiência, e por mais que as conversas iniciais com o cliente sejam esclarecedoras. Sem fazer o levantamento de requisitos, o que poderemos ofertar aos contratantes é apenas uma estimativa ou a técnica do "chutômetro". Pode-se até acertar, se o fator experiência, expertise do negócio e sorte, estiverem do seu lado. Mas, por mais veterano que você e sua equipe possam ser, o domínio do negócio a ser desenvolvido não é uma obrigatoriedade, e infelizmente o fator sorte é uma incógnita. Então, o que fazer?

Bom, quando disse que a resposta chegou, ela veio com o nome métrica. Resumindo, não há como estimar com boa aproximação, ou determinar o tamanho de um *software*, se não fizermos o uso de uma métrica. E para colaborar com o que digo, segue a citação de Drucker (2007) "A melhor estrutura não garantirá os resultados nem o rendimento. Mas a estrutura errada é a garantia do fracasso." Podemos entender que, caso não use uma métrica apropriada ou não use nenhuma métrica, a possibilidade de mensurar errado é muito alta, caso use a métrica certa, a possibilidade de acerto já passa a existir e num percentual aceitável.

Métrica pode ser: na literatura um conceito relacionado ao ritmo de um poema; na matemática um conceito ligado à distância; na música um conceito sobre a divisão de uma linha musical em compassos; e na Engenharia de *Software* (ESW) um conceito relacionado à gerência de projetos.

Bom, se estamos falando em engenharia, isto é, construções de algo não podem deixar de falar em medição, que é algo intimamente ligado a criar ou construir algo. E um *software* também precisa ser medido. Porém, essa é uma tarefa não muito fácil, já que estamos falando da construção de algo abstrato, que é um *software*, e assim sendo, possuindo um grau de subjetividade alto. E tudo que é subjetivo, está à mercê de interpretação as mais diversas, já que nesse caso, as referências, a experiência, entre outros fatores, influenciam a interpretação. Dessa forma, não existe ainda na ESW uma medição padrão aceito amplamente pela comunidade e que forneça resultados sem subjetividade. Isso tornar o processo de medir, muito árduo e muitas vezes, gerador de conflitos do tipo, o que medir o que avaliar, e se os resultados são aceitáveis.

Mas, quando fiz o curso de Análise de Pontos de Função (APF) o professor Guilherme Simões autor do livro citado na referência, que é certificado em AFP pelo IFPUG e utilizada essa métrica há muitos anos, nos disse que, ela não é a melhor métrica que existe, muito longe disso, mas em relação ao que existe hoje no mercado, ainda é a melhor. Assim, temos que ter consciência que medir o tamanho de um *software* é necessário, mas que qualquer métrica escolhida, terá suas limitações. Mas, sem uma métrica o processo de gerenciar a construção de um produto de *software*, planejando seu esforço,

seu custo, seus recursos, suas atividades seu prazo, etc., torna-se uma caixa preta cheia de surpresas desagradáveis.

As métricas na ESW são divididas em duas categorias, as quais são: Direta. Focada na medição do custo e do esforço de construção, isto é, tanto o desenvolvimento com a manutenção, seja ela evolutiva ou corretiva. Analisando a quantidade de Linhas de Código Fonte (SLOC) produzidas e o total de defeitos encontrados. Exemplos: custo, esforço, linhas de código, velocidade de execução, memória, número de erros, e complexidade ciclomática, isto é, a quantidade de caminhos de execução independentes a partir dum código fonte; e Indireta. Focada na qualidade e na funcionalidade do produto de *software*, em relação à aplicabilidade do *software*. Exemplos: funcionalidade, qualidade, complexidade, eficiência, confiabilidade, e manutenibilidade.

Essas ainda podem ser divididas em outras duas subcategorias, as quais são:

- Produtividade. Focada nas saídas do processo de engenharia de software;
- Qualidade. Focada no atendimento aos requisitos definidos pelo usuário.

Razões para Medir:

- Verificar a qualidade do software;
- Avaliar a produtividade dos desenvolvedores;
- Determinar as vantagens de novos métodos;
- Analisar as vantagens de novas ferramentas;
- Compor uma base para as estimativas;
- Fomentar oportunidades por refatoração;
- Justificar aquisições de recursos;
- Prever o custo do desenvolvimento;
- Estimar o prazo para entrega dos artefatos;
- Apoiar o planejamento do escopo do software;
- Mensurar os quantitativos de recursos; e
- Obter a produtividade dos recursos.

Métricas da Engenharia de *Software*:

- Pontos de Função (APF) que mede o tamanho funcional do *software*. Saber mais; e
- Pontos de Caso de Uso que mede o tamanho de um *software* a partir da utilização do mesmo pelos usuários. Saber mais.

1.1 Conceitos

Medida: A medição é a atividade básica de qualquer engenharia e não seria diferente para a engenharia de *software*, porém como esse campo é muito ressentido, está longe de desenvolver uma medição padrão que seja amplamente aceita e principalmente, tenha elementos para seus cálculos que não sejam baseados em fatores subjetivos. Consequentemente a comunidade de TIC possui muitas críticas sobre as medições hoje utilizadas e muita discordância sobre o que medir e como avaliar os resultados obtidos dessas medições.

Métrica: As métricas de *software* servem para realizar a tarefa fundamental do gerenciamento de projetos que é o planejamento. É com uma métrica que se pode identificar a quantidade de esforço necessário para o desenvolvimento, o custo para tal e o prazo para entrega dos artefatos. As métricas de *software* sob o ponto de vista de MEDIÇÃO são divididas em duas categorias: medidas diretas e indiretas.

Medidas Diretas: é composto por medições que objetivam descobrir o custo para produzir o *software*, o esforço necessário a ser aplicados ao desenvolvimento ou a manutenção do *software*, a quantidade de linhas de código produzidas e o total de *bugs* encontrados. Exemplos: Custo; Esforço; Linhas de Código; Velocidade de Execução; Memória; Número de *Bugs*; Complexidade Ciclométrica.

Medidas Indiretas: utilizam medições mais complexas, pois objetivam verificar a qualidade do *software*, a funcionalidade, ou a sua capacidade de manutenção, sendo mais difíceis de serem percebidas e avaliadas. Exemplos: Funcionalidade; Qualidade; Complexidade; Eficiência; Confiabilidade; Manutenibilidade.

As métricas de *software* sob o ponto de vista de APLICAÇÃO são divididas em duas categorias: medidas produtividade e qualidade. Medidas de Produtividade: verificam as saídas do processo de engenharia de *software*, e tem como objetivo de avaliar próprio processo de desenvolvimento. Medidas de Qualidade: verifica o quanto o *software* atende aos requisitos definidos pelo usuário, isto é, as funcionalidades assim permitem indicar o nível de resposta do *software* às exigências explícitas e implícitas do cliente, com relação ao que foi definido como qualidade.

1.2 Terminologias

MEDIDA: quantidade, dimensão, capacidade ou tamanho do *software*.

MEDIÇÃO: ato de medir.

INDICADOR: métrica que fornece compreensão dos resultados do *software*.

MÉTRICA: as técnicas utilizadas para verificar a funcionalidade, a modularidade, a manutenibilidade, etc., e podem ser subclassificadas como Orientadas ao Tamanho, Orientadas a Função, Orientadas a Pessoas.

1.3 Análise de Ponto de Função

Como dito anteriormente à métrica que mesmo não sendo a melhor, mas a que produz resultados menos conflitantes é a APF e é essa que vamos usar.

A Análise de Pontos de Função (APF) ou *Function Point Analysis* (FPA) foi criada em 1979 na IBM, tornando-se uma referência mundial na década de 80. A APF é um conjunto de técnicas que objetivam medir as funcionalidades, sendo assim, uma unidade de medida para fins práticos, não importando a tecnologia utilizada na construção do *software*. Resumindo, é uma métrica capaz de mensurar o tamanho do *software*.

O órgão responsável pela manutenção e divulgação das diretrizes da APF é o *International Function Point Users Group* (IFPUG), que publicou e mantém o Manual de Práticas de Contagem de Pontos de Função – *Counting Practices Manual* (CPM).

1.3.1 Cálculo do Ponto de Função

O objetivo da Análise de Pontos de Função (APF) é obter o tamanho de um *software* em Pontos de Função (PF) analisando as funcionalidades a serem desenvolvidas, isto é, os requisitos funcionais.

Porém é importante salientar que essa funcionalidade é sob o ponto de vista do usuário (PVU). Mas, o que quer dizer sob o ponto de vista do usuário? Significa que um Requisito Funcional (RF) tipo o que se segue RF001 deve ser analisado aos olhos do usuário e não do desenvolvedor.

RF001 - [...] é necessário armazenar os dados do cliente como CPF; nome; endereços residencial, comercial e de contato; telefones residencial, comercial, contato, celular; e-mails; sexo; nascimento; filiação; emprego atual; tempo de trabalho; emprego anterior; e faixa salarial [...].

Se um desenvolvedor fizer a leitura desse requisito, obviamente ele saberá que terá trabalho pela frente. Visto que, na sua visão tecnicista, já visualiza a modelagem do banco, que normalizará essa entrada em diversas tabelas, caso a persistência dos dados seja em um banco de dados relacional, e que uma série de classes de domínio, de controle, de persistência e de apresentação, terão que ser criadas, caso do desenvolvimento for numa linguagem orientada a objetos. Além da usabilidade, que exigirá uma interface gráfica (GUI) bem amigável, já que o cadastro poderá produzir várias visões para um único cadastrador.

Resumindo, sob ponto de vista do desenvolvedor (PVD), isso vai dar um trabalho danado. Porém, para o PVU se trata de um cadastro único. No qual serão cadastrados todos os dados acima levantados e simplesmente o utilizador irá clicar em um botão salvar e pronto. Para o PVU é muito simples. Qual o trabalho que tem isso? :)

Agora, por que se deve analisar pelo PVU? A resposta é: Para que a medição de um projeto de desenvolvimento de *software* e a manutenção deste não se preocupe com a tecnologia a ser utilizada na sua implementação e sim, apenas nas funcionalidades do sistema requisitadas pelo usuário. Explicando melhor, se você for desenvolver o RF001 em Clipper, ou em Pascal, ou em Delphi, ou Visual Basic, ou Java, ou CSharp, etc, o grau de esforço para cada linguagem é diferente, como também é diferente com o uso do tipo de arquitetura de banco de dados, se banco relacional é uma, se banco orientado a objetos é outra, assim a medição sofreria uma grande variação devido ao fator tecnologia empregada. Então, a mesma tem que ser isenta dessa preocupação, só focando nas necessidades do usuário.

1.3.2 Técnica

A primeira ação é determinar qual o tipo de contagem de pontos de função que deverá ser realizada. Podem ser de três tipos: Projeto de Desenvolvimento (PD); Aplicações Instaladas (AI); e Projetos de Manutenção (PM).

A segunda ação realiza a identificação do escopo de contagem, com a determinação da fronteira da aplicação, isto é, definir quais serão as funcionalidades que serão incluídas na contagem de PFs. Para isso a

definição da fronteira da aplicação é fundamental, pois estabelece as fronteiras entre aplicação, usuário e outras aplicações.

A terceira ação é determinar a contagem de Pontos de Função Não Ajustados (PFNA), isto é, verificar requisitos funcionais levantados, e classificá-los em dados e transações.

A quarta ação é a contagem dos dados. Para contar os dados se leva em consideração os dados *internos* os chamados Arquivos Lógicos Internos (ALI) e os dados *externos* os denominados Arquivos de Interface Externa (AIE) da aplicação. Ambos representam um grupo de dados logicamente relacionados que foram identificados pelo usuário. A diferença é que os ALIs são mantidos dentro da fronteira e os AIEs são os mantidos fora da fronteira da aplicação. Simplificando, os ALIs são persistidos pelas operações da aplicação, enquanto os AIEs são referenciados pela aplicação, mas não persistidos por ela.

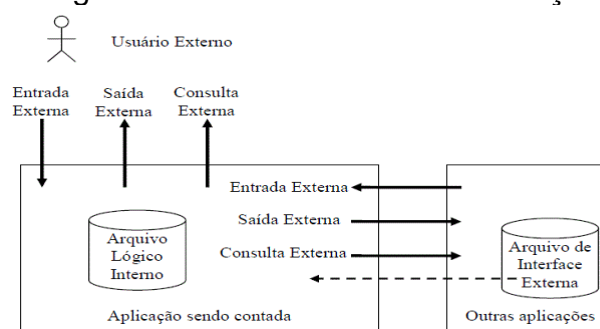
Entendido o que é uma métrica e o que se propõe a Análise de Ponto de Função, vamos aprender a contar PFs.

Exemplo, no RF001 terá que armazenar no banco de dados da aplicação os dados do cliente, e neste caso terá 1ALI, mas existe um requisito RF002 relacionado ao RF001 que diz:

RF002 - todos os endereços devem ser validados por CEP que deve ser consultado na base de dados dos correios.

Assim, para o referido cadastro do usuário, teremos 1AIE já que a persistência dos dados de CEP's é feita pelos correios que é outro sistema que se relaciona com o desenvolvido e está fora da fronteira da aplicação.

Figura 0.1 Análise de Ponto de Função



Fonte: Internet

A quinta ação consiste na contagem das transações. Para contar as transações se leva em consideração às funcionalidades de processamento de dados do sistema fornecidas para o usuário. Esses são classificados em Entradas Externas (EE), as Saídas Externas (SE) e as Consultas Externas (CE). As EEs são os processos responsáveis em manter os ALIs. As SEs são os processos responsáveis em exibir as informações recuperadas através de um processamento lógico, *com a realização de cálculos*. As CEs são os processos responsáveis em exibir as informações recuperadas através de um processamento lógico, *sem a realização de cálculos*.

A tabela a seguir define a complexidade dos ALI e os AIE:

Figura 1.2 Tabela Complexidade ALI e AIE

A L I A I E	TR TD	<20	20-50	>50
	1	Baixa	Baixa	Média
	2-5	Baixa	Média	Alta
	>5	Média	Alta	Alta

Fonte: IFPUG

A esta outra tabela define a complexidade do EE:

Figura 1.3 Tabela Complexidade EE

E E	TR TD	<5	5-15	>15
	<2	Baixa	Baixa	Média
	2	Baixa	Média	Alta
	>2	Média	Alta	Alta

Fonte: IFPUG

A próxima tabela define a complexidade dos SE e CE:

Figura 1.4 Tabela Complexidade SE e CE

S E C E	TR TD	<6	6-19	>19
	<2	Baixa	Baixa	Média
	2-3	Baixa	Média	Alta
	>3	Média	Alta	Alta

Fonte: IFPUG

A tabela ao lado é a define os pesos para a quantidade de ocorrências de dados e transações encontrados na análise dos requisitos funcionais:

Figura 1.5 Tabela de Contribuição

C o n t r i b u i ç ã o	Tipo	Baixa	Média	Alta
	ALI	7PF	10PF	15PF
	AIE	5PF	7PF	10PF
	EE	3PF	4PF	6PF
	SE	4PF	5PF	7PF
	CE	3PF	4PF	6PF

Fonte: IFPUG

A sexta ação serve para verificar as complexidades. Encontrados os ALIs, os AIEs, as EEs, as SEs e as CEs, devem-se verificar as suas complexidades funcionais e classificá-las em baixa, média ou alta. Essas complexidades são definidas com base em dois conceitos, os Tipos de Registros (TR) e os Tipos de Dados (TD). Os TRs correspondem ao conjunto de dados dentro de um ALI/AIE, que foram reconhecidos pelo usuário, exemplo, imagine o cadastramento de uma venda, teríamos no mínimo três tabelas de dados envolvidas no processo, a de vendas, itens-da-venda e a de produtos, mas todas as três seriam consideradas apenas 1RL. Os TDs são os campos reconhecidos pelo usuário como únicos e não repetidos, exemplo, o cadastro de cliente do RF001, temos o CPF, o nome, os fones comercial, residencial e celular, neste caso, temos 3ID's, já que os fones devem ser contados como

levar em consideração os componentes indiretos da métrica, isto é, as 14 características já citadas.

A nona ação deve determinar o valor do fator de ajuste, isto é, avaliar 14 características gerais de sistemas, que avaliam a funcionalidade geral da aplicação que está sendo contada e seus Graus de Influência (GI). O GI é determinado com base na seguinte escala: 0=Nenhuma influência; 1=Influência mínima; 2=Influência moderada; 3=Influência média; 4=Influência significativa; e 5=Influência forte.

A décima ação é aplicar o fator de ajuste de influência nos PFNA que fica na ordem de +/- 35%, aplicando-se as 14 características gerais dos sistemas, a saber: 1=Comunicação de Dados; 2=Processamento de Dados Distribuído; 3=Desempenho; 4=Utilização do Equipamento (Restrições de Recursos Computacionais); 5=Volume de Transações; 6=Entrada de Dados On-line; 7=Eficiência do Usuário Final (Usabilidade); 8=Atualização On-line; 9=Processamento Complexo; 10=Reusabilidade; 11=Facilidade de Implantação; 12=Facilidade Operacional (Inicialização, Cópia, Recuperação, etc.); 13=Múltiplos Locais e Organizações do Usuário; e 14=Facilidade de Mudanças (Manutenibilidade).

Para cada uma dessas 14 características deve-se atribuir um dos GIs para determinar o nível de influência, que indicará o quanto determinado à característica tem influência no sistema. Os 14 graus de influência (GIs) informados são somados, resultando no Nível de Influência Total (NIT).

A décima primeira ação consiste em determinar o Valor do Fator de Ajuste (VFA) pela fórmula $VFA = (NIT * 0,01) + 0,65$.

A décima segunda ação é o cálculo dos Pontos de Função Ajustados (PFA). Esse cálculo depende do tipo de projeto, se desenvolvimento, se manutenção ou se aplicações instaladas. Se for um projeto de desenvolvimento de sistemas então a fórmula é:

- Projeto_Desenvolvimento = Ponto_Função_Não_Ajustado * Valor_Fator_Ajuste;
- Aplicação_Implantada = Projeto_Desenvolvimento - (Pontos_Função_Não_Ajustados x Fator_Ajuste);
- Projeto_Manutenção = (Pontos_Função_Não_Ajustado + Ponto_Função_Incluído + Ponto_Função_Alterado_Atual - Ponto_Função_Alterado_Anterior - Ponto_Função_Excluído) x Fator_Ajuste_Atual.

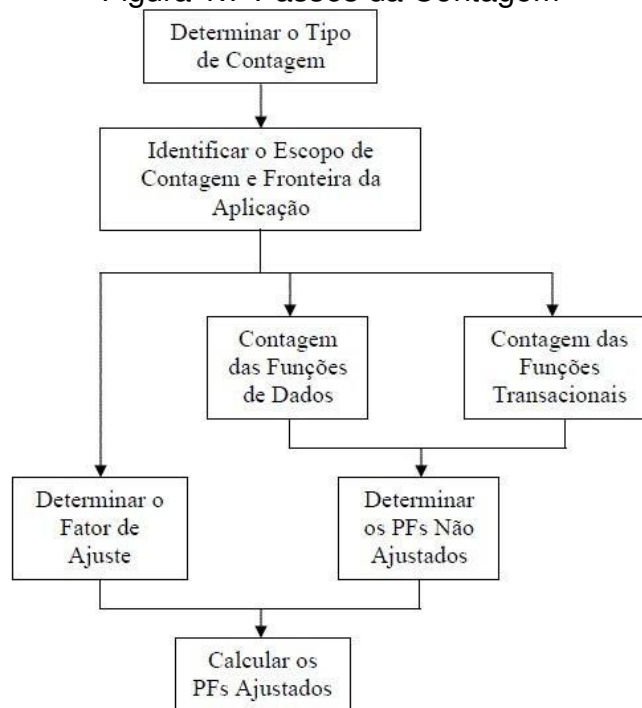
Com isso, se tem o tamanho do *software* e com esse tamanho (x PF's), pode-se determinar um valor a ser pago por cada ponto de função, e a partir dele determinar o valor total do *software*, além de determinar formas de pagamentos, relacionadas às entregas realizadas. Exemplo: se o tamanho do *software* é de 2.500 PFs e se paga R\$80,00 por todos os PF o valor total a ser pago é de R\$200.000,00. Mas, se no planejamento de entregas, ficou determinado que a contratada devesse entregar 50 PFs no mês e só entregou 25 PFs, o que será pago pela contratante a contratada, é o trabalho realizado, assim sendo, ela receberá R\$2.000,00 pelo trabalho feito, ficando os 25 PFs desse mês, para os próximos meses acrescidos dos outros 50PFs que já se

tenham planejado. Controle efetivo do que se tem a produzir e do que é produzido.

Clicando aqui se pode obter uma Tabela de Referência de PF para cada linguagem de programação.

Segue um resumo de todas as ações pode ser visto a figura a seguir.

Figura 1.7 Passos da Contagem



Fonte: Internet

Livros da série Fundamentos da Engenharia de Software



Fundamentos da Engenharia de Software: Conceitos Básicos é uma coletânea de disciplinas que integradas servem para fundamentar o

entendimento da construção de projetos de software com qualidade, isto é, baseado em processos maduros e reconhecidos pela comunidade tecnológica. O objetivo deste livro é fornecer ao leitor as bases necessárias para o desenvolvimento de aplicações sejam Desktop, Web ou Mobile. Iniciando a leitura na Teoria da Computação, passando por Processos, Linguagens, Bancos de Dados e finalizando com Sistemas de Informação e Colaboração. Este livro pode ser lido capítulo a capítulo ou somente a disciplina desejada, pois sua elaboração consiste na compilação das disciplinas fundamentais da Engenharia de Software que são independentes, mas ao mesmo tempo se integram objetivando o desenvolvimento de aplicações.



Introdução à Banco de Dados. Neste são abordados os conceitos básicos de bancos de dados e seus sistemas gerenciadores, mas com o foco

na arquitetura relacional, porque ainda hoje o mercado faz uso em larga escala desses bancos de dados, mesmo que o paradigma predominante seja o orientado a objetos e que, já existam a um bom tempo bancos orientados a objeto, até mesmo os bancos objetos-relacionais que são um híbrido entre essas duas arquiteturas, o que predomina ainda é o relacional, assim, este material é focado na linguagem de consulta estruturada para os SGBD-Rs do mercado, com foco na comparação de cinco dos mais utilizados bancos relacionais, os quais são: Oracle, SQLServer, MySQL, SQLBase e Interbase.



Este livro é sobre processos de desenvolvimento de software, evidenciando a necessidade de qualidade na construção de sistemas, conceituando a diferença entre desenvolvimento Adhoc e com processo. Para isso é realizado a introdução à engenharia de requisitos abordando as técnicas para a elicitação de requisitos que forneçam subsídios necessários para uma construção de software com maior qualidade, enfatizando a necessidade de se aplicar na construção de qualquer sistema as técnicas de análise e modelagem, evidenciando o uso da linguagem da Linguagem de Modelagem Unificada (UML) para diagramar um projeto de software, explicando a necessidade do uso de modelos na construção, entrando com detalhes na análise orientada a objetos, com o objetivo de explorar os seus conceitos de requisitos e modelagem integrados. Este material é finalizado com a introdução à medidas de esforço de desenvolvimento, técnica necessária para responder as perguntas básicas de qualquer desenvolvimento: Qual o prazo e custo? E para responder a essas questões é abordado o uso da métrica análise de ponto de função.



Este livro aborda os sistemas que são classificados como informação, a exemplo, sistemas de apoio a decisão, sistemas estratégicos, sistemas

gerenciais e sistemas transacionais. A produção deste material que compõe o volume 4 da coleção Fundamentos da Engenharia de Software é resultado da compilação das aulas produzidas nas disciplinas que compõem os capítulos deste livro.



A motivação deste livro é exemplificar os conceitos de Padrões de Projetos utilizando a linguagem de programação Java, sendo a construção uma

compilação das aulas produzidas com o intuito de facilitar o entendimento do assunto abordando os seguintes temas: Paradigma Orientado a Objetos que introduz o leitor nos conceitos do POO; Linguagem de Modelagem Unificada para apresentar a simbologia UML dos conceitos de POO; Linguagem de Programação Java apresentando essa poderosa linguagem de programação orientada a objetos para exemplificar os padrões de projeto; e Padrões de Projetos que neste livro aborda os mais referenciados nas academias, sendo eles o GRASP e GoF.



Este livro é o resultado do uso da ferramenta MS Project da Microsoft utilizada na aplicação dos conceitos de gestão de projetos do PMBOK com as premissas da engenharia de testes para aquisição de qualidade nos produtos de software.



desenvolvimento e lógica de programação.

Este livro aborda basicamente os conceitos básicos de programação como autômatos, tipos de linguagens, princípios dos compiladores, paradigmas de



dicas de codificação para a linguagem de marcação ASPX, para a linguagem de script mais utilizada pelos navegadores o JavaScript com exemplos de CSS e principalmente dicas de código para a linguagem de programação CSharp e de banco de dados SQL com foco no SQLServer.

Este livro introduz nas tecnologias Web abordando os conceitos básicos para desenvolvimento para Internet com a apresentação da plataforma Dot Net e exibindo