----------Príncipios de modularidade
1)Modulo
A)Def física: Unidade de compilação independente
B)Def Logica: Trata de um unico conceito

2)Abstração de sistema
Abstrair: Processo de considerar apenas o que é necessario numa situalão e descaretar com segurança o que não é necessario]
O res de uma abstração é um escopo

Nivel de abstração:
Sistem > Prog > Modulos > Funções > Blocos de Código > Linhas de codigo

OBS: Artefato - Um item de identidade propria criado dentro de um processo, coisas que não podem ser versionadas
Construto(build) - Primeiro resultado apresentavel, mesmo incompleto

3)Interface
Mecanismo de troca de dados, estados e eventos de um mesmo nivel de abstração
a)Exs de interface
>Arqv - Entre sis
>Func de acesso - Entre mod
>Passagem de param - Entre func
>Var globais - Entre blocos

B)Relacionamento cliente-server
(Desenhos feitos a parte)

C)Interface fornecida por terceiros
Depende de um terceiro componente para fazer dois modulos conversarem
 (Desenho a parte)

D)Interface em detalhe
>Sintaxe - Regras
>Semantica - Significado

E)Analise de interface
-Cliente interface: Ponteiro para dados validos ou NULL
-Server interface: Int válido
Ambos: Int ja conhecido

4)Processo de desenvolvimento
(Desenhos no caderno)

5)Bibliotecas estaticas e dinamica
-->Estatica:
>Vantagem
Lib ja acoplada
>Desvantagem
Existe uma copia dessa biblioteca estatica para cada exe na memoria
Aumenta o tamanho do programa
-->Dinamica:
>Vantagem
So carrega uma instancia de biblioteca dinamica na memoria
>Desvantagem:
a dll precisa estar na maquina

6)Modulo de definição - .h
>Interface do modulo

>Contem prototipos das funções de acesso
>Documentação voltada para o programador do mod cliente

7)Modulo de implementação - .c
>Codigo das func de acesso
>Codigos e prototipos da func internas
>Var internas

8)Tipo abstrato de dados (TAD)
Uma estrutura encapsulada em um modulo que somente é conhecida pelos mod clientes atraves de func de acesso na intertfa
Ex: Se um mod manipula uma matriz usa listas para fazer suas col e linhas
(Desenho caderno)

Se o cliente precisar usar mais e uma instancia da estrutura dada pelo TAD, uma solução é trabalhar com ponteiros para a cabeça da estrutura

9)Encapsulamento
>Objetivo:
Facilitar a manutenção
Impedir utilização ou modificação indevida de estrutura de mod
-->Outros tipos de encapsulamento:
    Documentação
>Documentação interna: Implementação .c
>Documentação externa: Definição .h
>Documentação de uso: README
    Codigo
>Blocos de codigos visiveis apenas no modulo ou dentro de outro bloco de codigos
>Codigo de uma função
    Variaveis
>Private, public,...

10)Acoplamento
Propriedade relacionada com a interface entre modulos
-->Conector: Item da interface
>Função de acesso
>Var global
-->Criterios de qualidade
>Quantidade de conectores
>Tamanho do conector
>Complexidade do conector

11)Coesão
Propriedade relacionada com o grau de interligação dos elementos que compõe o modulo
-->Niveis de coesão
>Incidental - pior
>Lógica - elementos logicamente relacionados
>Temporal - itens que funcionam em um mesmo periodo de tempo
>Procedural - itens em sequencia (ex: .bat)
>Funcional
>Abstração de tados - melhor coesão, trabalha um unico conceito (ex: TAD)

------------------------Teste Automatizado
1)Objetivo
Testar de forma automatica um modulo recebendo um conjunto de casos de teste na forma de um script e gerando um log de saida com a analise entre o que é esperado e o obtido
OBS:Apartir do primeiro retorno esperado diferente do obtido no log de saida todos os resultados de exec de casos de teste não são confiaveis

2)FRAMEWORK de teste
(Desenhos no caderno)

3)Script de teste
// - > comentario
== caso de teste - > testa determinada situação
= comando de teste ->associado a uma função de acesso
obs: teste completo -> caso de teste para todas as condiçoes de retorno de cada função de acesso do modulo

4)Log de saida
== caso 1
== caso 2
== caso 3
1>> função espera 0 e retorna 1
0<< Recuperar

5)Parte especifica
A parte especifica que necessita ser implementada para que o framework(arcabouço) possa acoplar na aplicação chama-se hotspot
ex: testarv.c

```
#define criar_arv CMD "=Criar"
efetuar comando(comando teste)
    if(strcmp(comando teste,Criar Arv CMD)
        numlidos = ler_leparametros("i", &com ret esperada)
```

obs: ret esperada = 0

```
    if(Numlidos != 1)
        return TST_Condret
    Cond ret obtido = Arv_grav
    Cont ret obtida //Erro ao criar arv
```

--------------------------Processo de desenvolvimento em engenharia de software

(Desenho caderno)
->Lider do projeto
->Gerencia de configuração
->Qualidade de software

(Desenho caderno)

Requisitos:
>Elicitação: Entrar em contato com o cliente e ver o que é necessario
>Documentação
>Verificação
>Validação

Analise de projeto:
>Projeto logico
>Projeto fisico

Implementação:
>Programas
>Concretiza a linguagem de programação
>Teste unitário

Testes:
>Teste integrado

Homologação:
>Sugestão: não estava nas especificações, adicionar especificações
>Erro: estava nas especificações, a equipe não cumpriu