Projeto de Sistemas I

7.1 - Análise de Transformação
7.1 - Análise de Transação

1 - Introdução

1.1 - Proposta para o Projeto

Projeto estruturado de Sistemas é a atividade de transformação das atividades do usuário, provenientes da fase de Análise, em um plano de implementações através da automação eletrônica. Em outras palavras, é a passagem do "o que" para o "como" fazer.

Nesta fase, deixa-se de ter a tecnologia perfeita, passando a levar em consideração o hardware e o software com todas as suas limitações.

A fase de projeto procura prover os sistemas de facilidades para:

- Construção
- Teste
- Modificações
- Entendimento

Essas metas são alcançadas cumprindo-se os seguintes aspectos :

- Permite que a forma do problema guie a forma da solução
- Procura a resolução da complexidade dos grandes sistemas através da segmentação de um sistemas em "caixas pretas", e pela organização dessas "caixas pretas" em uma hierarquia conveniente para implementações computadorizadas.
- Utiliza ferramentas, especialmente gráficas, para tornar os sistemas de fácil construção compreensão e manutenção.
- Oferece um conjunto de estratégias para desenvolver o projeto de solução de uma declaração bem definida do problema
- Oferece um conjunto de critérios para avaliar a qualidade de um dado projeto-solução com respeito ao problema resolvido

1.2 - Critérios de Qualidade do Projeto

- Manutenabilidade

Deve-se projetar o sistema de forma a permitir facilidades de alteração devido a:

a. erros no sistema
b. novas necessidades do usuário
c. alterações do ambiente

Verifica-se que os sistemas mais fáceis de alterar são aqueles construídos de forma modular, com cada módulo desempenhando funções bem definidas e coesas, e com baixo grau de interdependência ou acoplamento entre os mesmos.

• Performance

Performance está diretamente relacionada ao uso otimizado dos recursos de hardware, software e pessoal disponíveis.

Como parâmetros de avaliação da Performance, temos:

a. tempo de procesamento

b. memória ocupada

c. through-put - dados processados por unidades de tempo

d. tempo de resposta

Dentre os fatores que afetam o desempenho, temos:

a. defiência do projeto de interface

b. tempo de acesso a discos e periféricos - devido ao tempo gasto em acesso a discos ser muito maior do que as operações na CPU, deve-se prover sistemas de mecanismos que minimizem este tempo, empregando-se organizações de arquivos adequadas.

c. falta de reogarnização de arquivos - em arquivos grandes e de muita utilização, deve-se verificar a possibilidade de limpezas periódicas, ou particionamento do mesmo gerando um arquivo atual e um histórico.

d. processos longos em horários de pique - deve-se evitar ao máximo a execução de processos batch de longa duração durante o horário de pique, tais como transferências de grandes arquivos, atualizações de grande volume de dados em banco de dados.

• Segurança

Deve-se prover mecanismos para evitar acessos indevidos ao sistema.

Como tipos de ameaças a segurança temos:

a. infiltração intencional - roubo, sabotagem, grampeamento

b. infiltração não intencional - linhas cruzadas, irradiações

c. acidentes - erro do usuário, falha de hardware, falha de software

Para controle de acesso ao ambiente deve identificar usuário e terminal, restringindo as operações conforme a necessidade.

Cada usuário deve possuir um único e pessoal:

a. código de identificação - usuário

b. código de autenticação - password

• Interatibilidade

Corresponde a facilidade do usuário em interagir com o sistema.

Cuidados com lay-out de telas e ralatórios devem ser observados. As telas devem ser padronizadas para menu; atualização, consulta; entradas de parâmetros; espera de processamentos longos e respostas de consultas. As teclas de funções válidas em cada tela devem ser indicadas explicitamente.

Telas devem ser providas de:

a. data
b. identificação do usuário
c. código da tela
d. identificação da operação referente a tela
e. área de menu de opções e entradas de dados
f. área de mensagens de erro relativos às operações
g. área de mensagens interativas como decorrer da operação
h. Indicação das teclas de funções

A crítica de campos pode ser feita campo a campo, tela cheia, ou por grupos de campos. Mensagens de erro devem ser padronizadas e devem deixar claro qual foi o erro, e a indicação do campo, se necessário. Deve-se prover DEFAULTS para entradas padrões e utilização de teclas para acesso a tabelas em campos codificados

Linha de comando deve ser utilizada principalmente para sistemas com um alto nível hierárquico de telas. Este recurso permite que usuário através de um mneumônico de uma aplicação, tenham acesso a mesma sem a necessidade de navegação pelas telas.

• Confiabilidade

Confiabilidade está relacionada a redução do risco de interrupção no fluxo normal de processamento das informções causadas por:

a. indisponibilidade de acesso - o sistema não oferece o serviço no tempo estipulado ou desejado.
b. perda da integridade das informações

Como medida para aumentar a confiabilidade temos:

a. restrições de integridade que podem ser feitas via programas ou definidas no próprio SGBD, quando este possui esta facilidade;
b. crítica e entrada de dados;
c. programas de acertos que geram ajustes, estornos, cancelamento ou correções;

Mecanismos de recuperação de falhas devem ser utilizados de acordo com o nível de confiabilidade da aplicação, tais como back-up e Log

- Economia

O custo do projeto deve ser balanceado de acordo com:
a. custo da tecnologia
b. custo operacional
c. custo de construção e manutenção

4 - Ferramentas do Projeto Estruturado

Análogo a todas as disciplinas de engenharia, o desenvolvimento de sistemas computadorizados requer o uso de ferramentas. Mas projeto de sistemas bem-sucedidos necessita escolher a ferramenta apropriada para um propósito específico.

4.1 - Diagrama de Estrutura

A idéia de segmentar o sistema em "Caixas Pretas" **(Modularização)** elimina a necessidade de conhecer como ela trabalha internamente para podermos utiliza-la, pois sua complexidade poderá ser melhor estudada posteriormente, evitando assim a impossibilidade na compreensão de grandes sistemas pela sua complexidade e elevado número de módulos.

As características das caixas pretas são:

1. Você conhece como devem ser os elementos na entrada.
2. Você sabe são os elementos na saída.
3. Você conhece sua Função (o que a caixa preta faz para que, com as entradas sejam produzidas as saídas.
4. Você não precisa conhecer como ela realiza suas operações para poder utiliza-la.

O diagrama de Estrutura ilustra a segmentação de um sistema em **módulos**, mostrando sua **hierarquia**, **organização** e **comunicação**. Com isto ele reduz a complexidade, facilita as alterações e permite a reutilização de código dos sistemas.

Com a utilização do diagrama de estrutura é mais fácil entender, programar, consertar, testar e atualizar os sistemas. Pois:

- Cada módulo pode ser implementado independentemente.
- Há menor possibilidade de erros na manipulação de menos código.
- Os efeitos colaterais das alterações são minimizados, pois podemos identificar a priori quais os módulos poderão ser afetados.
- Os esforços de otimização podem ser reduzidos aos pontos mais críticos.
- Módulos de função única podem ser reutilizados em outros sistemas.
- As mudanças de pessoal são menos críticas.

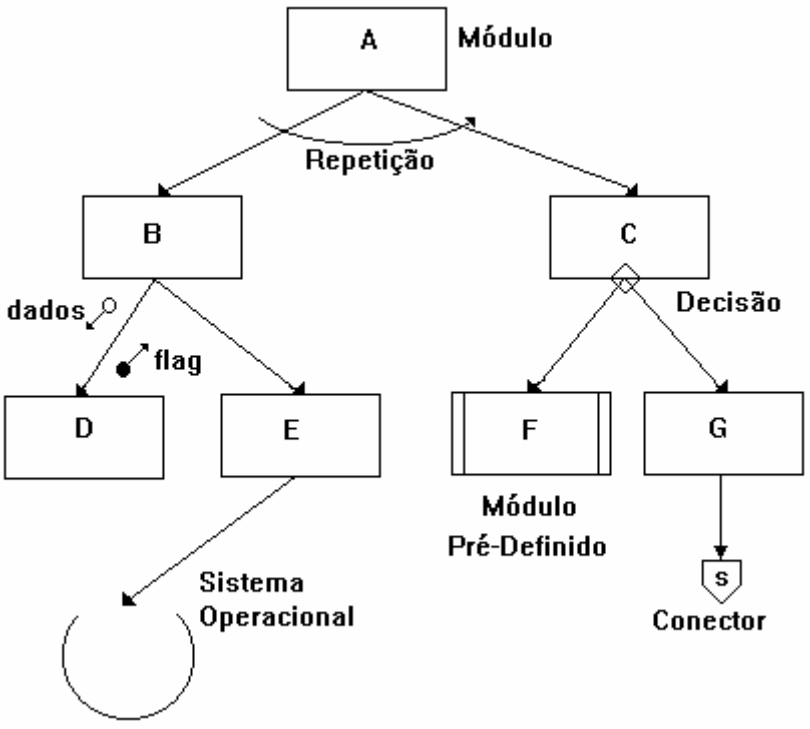
Componentes do Diagrama de Estrutura.

- Módulo - uma coleção de instruções de programa com seis atributos básicos: nome, entrada, saída, função, lógica, dados internos

1. Nome - todo módulo precisa ter uma identificação, para que possa ser referenciado no sistema.
2. Entrada - dados a serem manipulados para produzir a saída.
3. Saída - dados produzidos com o processamento da entrada.
4. Função - o que faz com a entrada para produzir a saída.
5. Lógica - a lógica com a qual ele executa sua função.
6. Dados Internos - seu espaço privado de trabalho, dados para os quais só ele faz referências **(encapsulamento).**

Símbolos do Diagrama de Estrutura

- O módulo é representado por um retângulo com seu nome no interior.
- Um módulo pode ativar (chamar) outro módulo, esta representação é feita por uma seta sempre direcionada na direção da ativação (para baixo).
- Um módulo pode enviar ou receber **dados** e/ou **controles** para outro módulo, esta representação é feita através de setas com bolinhas acompanhadas do nome do dado. Quando os parâmetros forem dados as bolinhas serão vazias e quando forem controles (Flags) as bolinha serão preenchidas.
- Um módulo pode chamar outro de forma condicional, através do uso de um losango.
- Um módulo pode chamar outro módulo repetidas vezes, através de um semicírculo.
- Um módulo pode já estar pré-definido
- Um círculo incompleto representa o Sistema Operacional



OBS: Os símbolos do Diagrama de Estrutura representam os mesmos comandos da Programação Estruturada: seqüência, seleção e repetição.

4.2 - Pseudocódigo

O PseudoCódigo é uma linguagem de especificação informal muito flexível que não tem a intenção de ser executada em uma máquina, mas pode ser usada na organização dos pensamentos dos programadores antes da codificação. Apesar do pseudocódigo ser uma ferramenta de programação estruturada ela pode também ser utilizadas em projetos estruturados para esclarecer a rotina detalhada de procedimentos de algumas caixas pretas do diagrama de estrutura.

5 - Atividades da Fase de Projeto Estruturado

O Diagrama de Estrutura mostra somente a estrutura geral de um sistema e deliberadamente reprime quase todos os detalhes de procedimento. No entanto, o programador que usará o diagrama de estrutura para derivar sua codificação, está muito interessado em detalhes de procedimento. Como então essas informações podem ser fornecidas ao programador?

O Projeto Estruturado não se resume ao Diagrama de Estrutura, pois este apresenta um alto grau de abstração. Para servir como uma boa entrada a fase de implementação, é preciso que mais algumas coisas sejam detalhadas além do Projeto da arquitetura, tais como: Projeto Detalhado, Projeto de Arquivos e Projeto da Interface com o Usuário.

5.1 - Projeto da Arquitetura

- Especifica a Organização Modular Lógica, definindo a estrutura organizacional do Sistema. Mostra a hierarquia dos Módulos, suas dependências e necessidades de informação.
- Define controles, mostrando quais as condições de funcionamento de um determinado módulo
- Nesta etapa só é relevante definir a função módulo, neste ponto só interessa mostrar o que o módulo faz e não como ele faz (abstração). Neste ponto ainda não se estar dependente de hardware ou software.
- Nesta etapa a ferramenta utilizada é o Diagrama de Estrutura.

5.2 - Projeto Detalhado

- Especifica os algoritmos que implementam as funções identificadas na organização modular.
- Neste ponto a estrutura hierárquica definida no projeto da arquitetura é detalhada de forma que os módulos passem a mostrar como eles executarão suas funções.
- Nesta etapa a ferramenta mais utilizada é o PseudoCódigo.

Obs: Desta etapa em diante as descrições já estão dependentes de hardware e Software.

5.3 - Projeto de Arquivos

- Definição do conteúdo dos arquivos, das características físicas de armazenamento dos dados e do acesso aos mesmos.
- Na especificação de requisitos foram levantadas as necessidades de informação. Naquele momento não importava como estas informações seriam armazenadas, pois não haviam restrições tecnológicas que pudessem impor algum tipo de armazenamento. Entretanto, neste momento, precisamos definir como estas informações serão armazenadas
- Nesta etapa a ferramenta utilizada fica a critério do projetista, podendo utilizar um dicionário de Dados.

5.4 - Projeto da Interface do Usuário

- Definição dos mecanismos de interação do usuário com o produto
- Neste ponto iremos definir como será a forma de comunicação do usuário com o produto. Este ponto é crítico para o projeto e pode determinar o sucesso ou total fracasso do produto.
- Nesta etapa é conveniente utilizar uma ferramenta gráfica para mostrar a priori como será a interface, com o usuário, do sistema.

Obs: Independente das ferramentas escolhidas pelo projetista para especificar um módulo para o programador, essas ferramentas não podem prescindir da boa comunicação humana. Se você construir paredes entre as pessoas, você terá sérios problemas. Não separe o grupo de programação do grupo de projeto; deixe que os programadores tenham acesso ao diagrama de estrutura mesmo antes que ele esteja terminado. De outra forma haverá desentendimento, não importando quão boas sejam as ferramentas de especificação.

6 - As Qualidades de Bom Projeto

Nem o Diagrama de estrutura, nem as especificações de módulos por si só nos dizem muito a respeito das qualidades de um projeto. O diagrama estruturado é simplesmente uma ferramenta que mostra os módulos de um sistema e seus relacionamentos, mas como poderemos identificar se estes módulos e estes relacionamentos tem boa qualidade?

Um dos princípios fundamentais do projeto estruturado é que um grande sistema deveria ser particionado em módulos "mais" simplificados. No entanto, é vital que essa partição seja feita de tal maneira que os módulos sejam tão independentes quanto possível (Acoplamento) e que cada módulo execute uma única função (Coesão).

6.1 - Acoplamento

O acoplamento mede o grau de dependência entre os módulos de um sistema, devemos então reduzir ao máximo o acoplamento. Existem várias formas e intensidade de acoplamento entre módulos, sendo eles em ordem crescente de intensidade:

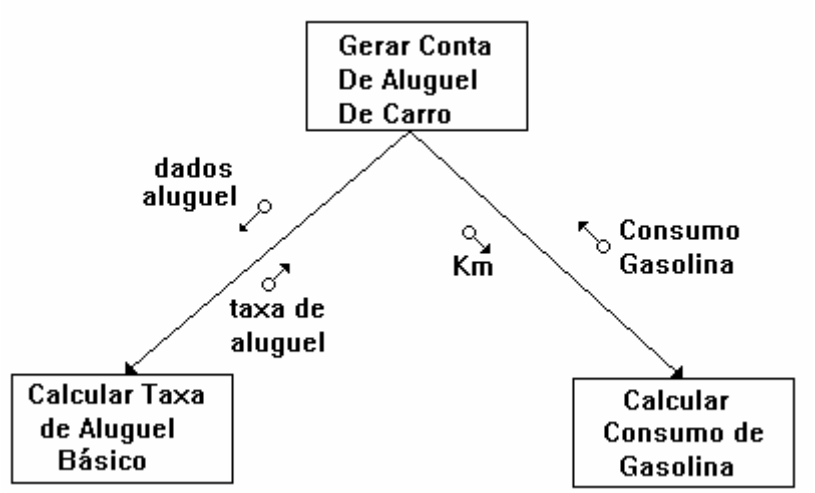
1. Acoplamento de Dados
2. Acoplamento de Imagem

3. Acoplamento de Controle
4. Acoplamento Comum
5 .Acoplamento de Conteúdo

1. Acoplamento de Dados

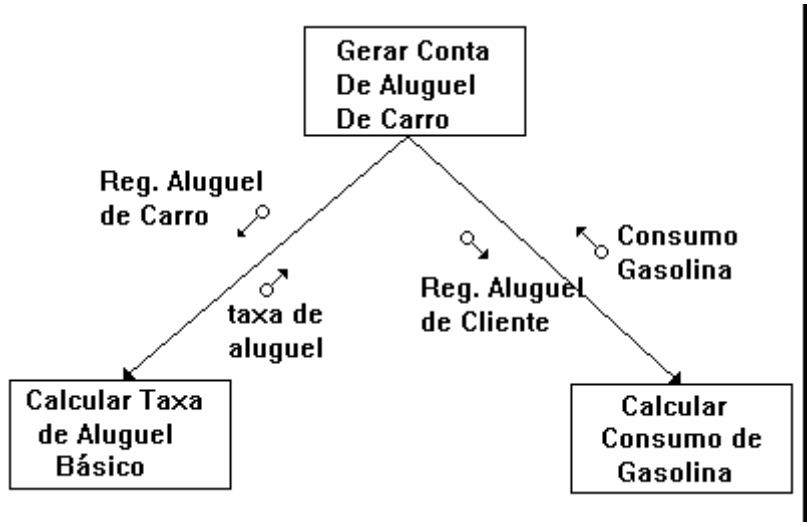
Dois módulos possuem este tipo de acoplamento quando eles se comunicam via um item comum de dados (parâmetro único ou uma tabela homogênea) que é passado diretamente entre eles.

É a comunicação de dados necessária entre módulos.



2. Acoplamento de Imagem

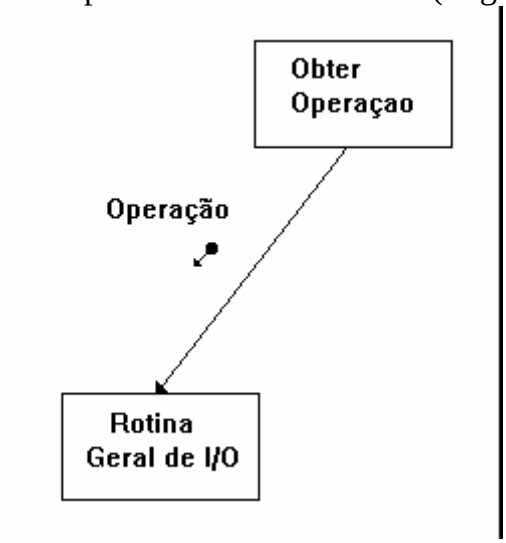
Dois módulos são ligados por imagem se eles se referem à mesma estrutura de dados.



Neste caso a estrutura de dados Reg. Aluguel de Cliente está sendo passada para o módulo Calcular Consumo de Gasolina, mesmo que este não precise de toda a estrutura para efetuar sua função. Com isto qualquer alteração na estrutura de dados irá causar problemas no módulo, mesmo este não precisando de toda a estrutura.

3. Acoplamento de Controle

Dois módulos possuem este tipo de acoplamento quando os dados de um são utilizados para definir a ordem de execução das instruções de outro módulo . Isto acontece quando um flag é passado para um módulo inferior (flag de controle).



4. Acoplamento Comum

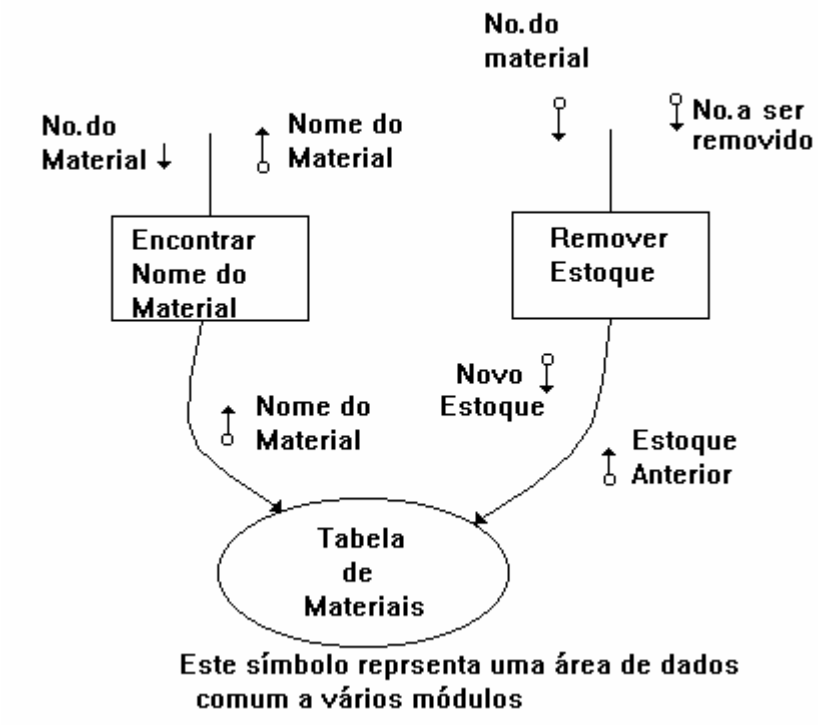
Este tipo de acoplamento ocorre quando mais de um módulo fazem referência a uma mesma área de dados, causando sérios problemas como:

1 - Um erro em qualquer módulo que use uma área global pode causar resultados indesejáveis em qualquer outro módulo que utilize a mesma área global.

2 - Inconsistência de dados.

3 - Dificuldade de compreensão dos módulos já que as variáveis estão definidas em outro local.

4 - Dificuldade em descobrir quais dados devem ser alterados se um determinado módulo for alterado e também quais módulos devem ser alterados se determinado dado for alterado.



5 .Acoplamento de Conteúdo

Ocorre quando um módulo faz referência ao interior do outro (quebra o encapsulamento de código). Imaginem uma função utilizando um **GOTO** para o interior de outra função.

6.2 - Coesão

Mede o grau de relação entre as atividades "**comandos**" de um módulo, quando maior a coesão mais fácil será a reutilização do módulo. Os tipos de coesão são em ordem decrescente de intensidade:

1. Funcional
2. Seqüencial
3. Comunicacional
4. Procedural
5. Temporal
6. Lógica
7. Coincidental

Onde,

1. Funcional

Um módulo com coesão funcional contém elementos que contribuem para execução de uma e apenas uma tarefa relacionada ao problema. Este tipo de coesão é mais encontrado nos níveis inferiores do diagrama de estrutura, mas caso os módulos tenham sido bem definidos pode-se encontrar este tipo de coesão em todos os níveis do diagrama, pois não importa a complexidade ou quantas sub-funções o módulo contém - se for possível resumi-las em uma função relacionada ao problema, então o módulo é funcionalmente coeso.

Ex: CalcularSalário, ImprimirCabeçalho, etc.

2. Seqüencial

Este tipo de coesão ocorre quando os elementos de um módulo estão envolvidos em atividades tais, que os dados de saída de uma atividade servem de entrada para próxima.

```
Ex:   LerSalarioBruto;
      CalcularDesconto(SalárioBruto);
      CalcularSalárioLíquido(SalárioBruto, Desconto);
```

3. Comunicacional

Quando os elementos de um módulo operam todos sobre os mesmos dados.

Onde, este módulo poderia ser transformado em dois módulos:
ProduzirRelatórioDeSaláriosDeFuncionários;
CalcularMédiaDeSalários;

Obs: Num DFD estes módulos seriam processos recebendo o mesmo fluxo de dados.

4. Procedural

Deste tipo de coesão em diante cruzamos as fronteiras dos módulos de fácil manutenção. Este tipo de coesão ocorre quando os elementos de um módulo estão envolvidos em atividades diferentes e possivelmente não relacionadas, nas quais o controle flui de uma atividade para outra, a diferença entre este tipo de coesão e a coesão seqüencial, é que, na coesão seqüencial são os dados que fluem de uma atividade para outra e não o controle.

Ex:
```
AbrirArquivo;
PesquisarChave;
LerRegistro;
GravarAumentoSalário;
ImprimirRegistro;
FecharArquivo;
```

5. Temporal

Um módulo com coesão temporal é aquele cujos elementos estão envolvidos em atividades relacionadas no tempo. Um bom exemplo de coesão temporal são as rotinas de inicialização.

Ex:
```
DeclararVariáveis;
AbrirAquivos;
ConfigurarPeriféricos;
IndexarArquivos;
```

6. Lógica

Um módulo logicamente coeso é aquele cujos elementos contribuem para atividades da mesma categoria geral, onde a atividade ou atividades a serem executadas são selecionadas fora do módulo.

Ex:
```
Ler(Arquivo, Operação)
Se Operação = 1
        Imprima(Arquivo);
Se Operação = 2
        Apague(Arquivo);
Se Operação = 3
        Copie(Arquivo);
FimSe;
```

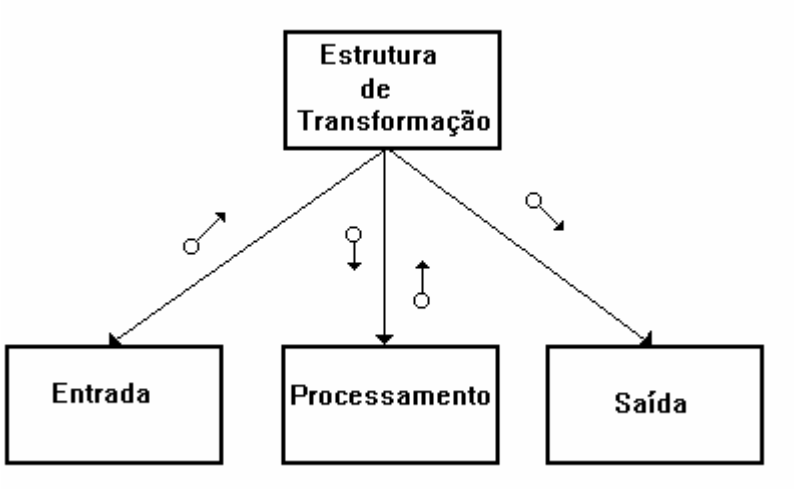
7. Coincidental

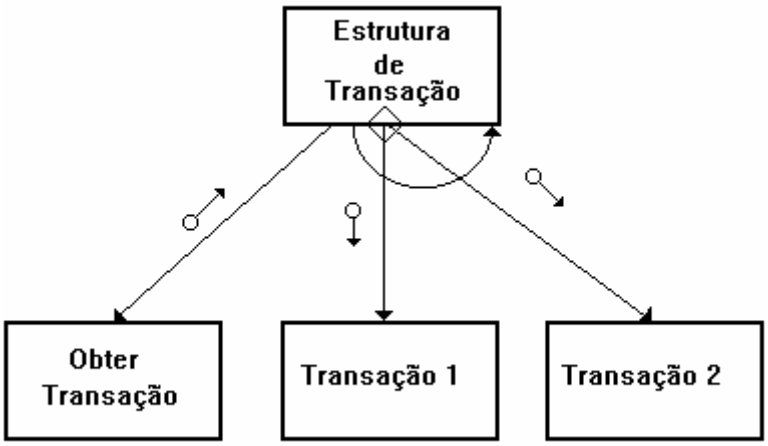
Ocorre quando os elementos de um módulo não são relacionados por qualquer função, procedimento, dados ou qualquer outro fator. Eles "tinham de ser colocados em algum lugar!".

Ex:
```
A = 7;
B = C*E;
Leia(x);
```

7 - Construindo um Diagrama de Estrutura a partir de um DFD

Como o Projeto Estruturado é fase na qual descrevermos a solução do problema através do estudo do problema (fase de análise), devemos de alguma forma transformar a saída da fase de análise na entrada da fase de projeto. Como a principal ferramenta da análise estruturada, que retrata o problema estudado, é o DFD devemos encontrar uma forma de passar "O Problema" para "A Solução", esta forma seria transformar o DFD em Diagrama de Estrutura. Isto é possível através do uso das estratégias de projeto: Análise de Transformação e Análise de Transação:





7.1 - Análise de Transformação

Análise de Transformação (Projeto Centralizado de Transformação) é a principal estratégia para transcrever o DFD da Análise em Diagrama de Estrutura do Projeto. Tendo em mente que esta é parte mais delicada do ciclo de vida de um Sistema, é importante que algumas regras sejam seguidas, mas ainda assim a principal ferramenta do projetista é a comunicação com os analistas, afim de ter certeza que o diagrama de estrutura esta representando a solução do problema, senão todo o esforço no desenvolvimento do sistema será em vão. Aqui vão alguns passos que podem ser utilizados nesta transformação:

1 - Identificar o Centro de Transformação - local onde as entradas são transformadas em saídas. Esta tarefa não é trivial e quase sempre devemos usar o bom senso e a observação. Uma boa forma seria percorrer todo o DFD e identificar os fluxos nos quais os dados de entrada sofrem pequenas alterações (refinamentos), em seguida identificar os fluxos de dados que representam a saída não formatada e então observar, que o centro de transformação está entre estes fluxos. É interessante observar que nas aplicações comerciais o centro de transformação é composto por uma pequena parcela do DFD.

2 - Desenhar o Diagrama de Estrutura - Identificado o centro de Transformação devemos então colocar os processos, que o representam, num nível superior, as entradas do lado esquerdo e as saídas no lado direito. Então a partir daí, devemos nomear os módulos do diagrama de estrutura representado as funções do DFD (Entrada, Processamento e Saída). Nomes de módulos necessariamente não são iguais aos nomes de processos pela diferença de enfoque dos Módulos e dos Processos, um Processo descreve apenas sua função, enquanto um módulo representa o conjunto de atividades de seus subordinados.

3 - Revisar o Diagrama de Estrutura - Para ter certeza do sucesso do diagrama de estrutura é necessário que este passe pelas mãos da equipe de análise e sofra algumas revisões. Uma boa forma de conferir o diagrama de estrutura é analisa-lo junto com o DFD, afim de verificar se este implementa corretamente os requerimentos do DFD.

7.1 - Análise de Transação - A análise de transação tem duas funções principais: primeiro, para dividir um DFD de grande porte e alto grau de complexidade em DFDs menores, um para cada transação que o sistema processa. Segundo, análise de transação pode ser usada para combinar os diagramas de estruturas individuais de transações separadas em um diagrama de estrutura maior e mais flexível em relação às mudanças do usuário.

Uma transação, de forma geral, é um estímulo para um sistema que possui um conjunto de atividades a serem realizadas internamente. A análise transformação é na verdade uma extensão no conceito de modularização, pois ela separa as transações em módulos distintos e mais compreensíveis, que podem ser relacionados através de um módulo superior que define qual transação será executada. Um bom exemplo seria um Menu, onde temos as opções: Incluir, Alterar, Excluir. O menu seria a estrutura de Transação, o qual de uma forma natural é largamente utilizado para modularizar as transações de um sistema.