

# Engenharia de Software

## Ficha T. Prática nº 4

*Fonte: Eng. De Software, Coleção Schaum*

### **Objectivo:** Planeamento do Projecto de Software

#### **1. Planeamento de software**

O planeamento é essencial, e o desenvolvimento de software não é a excepção. Alcançar sucesso no desenvolvimento de software requer planeamento, o qual neste caso, envolve decidir quais tarefas precisam ser feitas, em que ordem fazê-las e que recursos são necessários para cumpri-las.

#### **PARTE I**

#### **2. Estrutura de decomposição do trabalho**

Uma das actividades essenciais do planeamento, é a decomposição das grandes tarefas em pequenas tarefas. Isso significa encontrar partes identificáveis das tarefas, que podem ser usados para estimar o esforço requerido e medir o progresso.

A estrutura de decomposição de trabalho (work breakdown structure – WBS) deve ser uma estrutura de árvore.

A seguir estão as regras para construir uma estrutura de decomposição do trabalho:

- A WBS deve ser uma estrutura de árvore.
- Toda tarefa e a sua descrição devem ser compreendidas e não ambíguas. O propósito da WBS é a comunicação entre os membros da equipa. Se membros da equipa interpretarem mal o que a tarefa ou documento sugere fazer, haverá problemas.
- Toda tarefa deve ter um critério de conclusão . Deve existir umamaneira de decidir quando uma tarfa está completa, porque subtarefas que não possuem um final definido incentivam expectativas de falso progresso. Esta decisão é chamada critério de conclusão (estamos a falar em milestones, ou entregáveis)
- Todos os artefactos devem ser identificados. Um artefacto deve ser produzido por uma tarefa específica, caso contrário não será produzido.

#### **3. WBS de “fazer pão”**

- Seleccionar ingredientes
- Misturar ingredientes
- Cozinhar
- Comer
- Limpar

#### **4. Decompondo em subtarefas e indicando os entregáveis**

<b>Sub-tarefa</b>	<b>Entregável</b>	<b>Esforço</b>
Escolher ingredientes	Lista de ingredientes	2 pessoa-hora
Comprar ingredientes	Ingredientes comprados	2 pessoa-hora
Misturar ingredientes	Farinha	0,2 pessoa-hora (12 min)
Adicionar liquidos	Líquidos na tigela	0,1 pessoa-hora (6 min)
Adicionar fermento	Líquidos com fermento	0,1 pessoa-hora (6 min)
Adicionar um pouco de farinha	Líquido com farinha	0,1 pessoa-hora (6 min)

Deixar crescer a 1ª vez	Mistura a crescer	0,5 pessoa-hora (30 min)
Adicionar a farinha restante	Massa totalmente misturada	0,2 pessoa-hora (12 min)
Deixar crescer a 2ª vez	Massa a crescer	1 pessoa-hora (60 min)
Colocar nas formas de pão	Pães crus	0,25 pessoa-hora (15 min)
Cozinhar (Assar)	Pão	0,25 pessoa-hora (15 min)
Fatiar	Fatias de pão	0,25 pessoa-hora (15 min)
Passar manteiga	Fatias com manteiga	0,1 pessoa-hora (6 min)
Comer	Pão comido	0,20 pessoa-hora (12 min)
Limpar	Cozinha limpa	0,50 pessoa-hora (30 min)

## 5. WBS dum software de visão

A equipa XYZ deseja desenvolver um sistema de reconhecimento de face para usar num robot. O sistema pretende receber visitantes no laboratório de robótica. Ele deve reconhecer faces que já tenha visto com uma razoável fiabilidade. No primeiro passo da decomposição de tarefas reconhece-se as primeiras tarefas

### Possibilidades

- Possibilidades da visão

- Determinar disponibilidade de câmara e software

- Planear aquisição de câmara e software de reconhecimento

### Fazer análise de risco

### Especificar requisitos

### Elaborar protótipos

### Implementação

- Codificar a captura de imagem

- Codificar o processamento da imagem

- Codificar a comparação da imagem

- Integrar com outros software do robot

### Testar a captura de imagem

### Documentação

Várias tarefas estão num nível que impede a estimação do esforço e duração requerida. Portanto devem ser detalhadas ainda mais. Segue a decomposição da tarefa de codificação da captura da imagem

Tarefa	Entregável	Esforço
Instalar um dispositivo de câmara comercial	Dispositivo instalado	1 pessoa-dia
Testar dispositivo a partir do windows e salvar a imagem no arquivo	Arquivo de imagem	2 pessoas-dia
Escrever rotina para chamar o dispositivo	Rotina escrita	15 pessoas-dia
Testar rotina separadamente e salvar imagem no arquivo	Imagem no código	7 pessoas-dia
Testar rotina do software de controlo do robot principal e capturar a imagem	Imagem principal	7 pessoas-dia

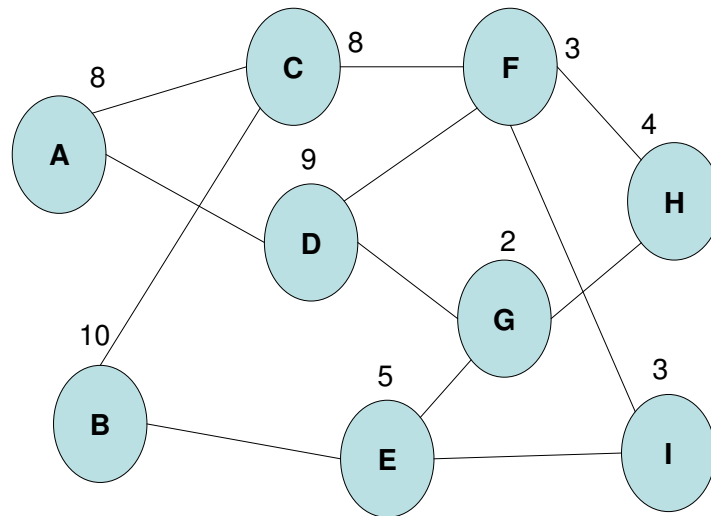
## PARTE II

### 6. PERT

Esta técnica cria um grafo que mostra as dependências entre as tarefas. Cada tarefa tem uma estimativa de tempo necessária para completar a tarefa e uma lista de outras tarefas que têm que ser completadas antes que esta tarefa possa ser iniciada (dependências). O grafo pode não ter sempre uma única subtarefa ou apenas uma tarefa de término. A tarefa completa é encerrada quando TODAS as suas tarefas forem encerradas. O grafo pode ser usado para calcular os tempos de conclusão para todas as subtarefas, o tempo mínimo de conclusão para a tarefa total e o caminho crítico das subtarefas.

**Tempo de conclusão:** para calcular o tempo de conclusão deve somar o tempo para completar cada tarefa ao tempo requerido para completar todas as suas dependências. Comece sempre pelas tarefas que não têm dependências, já que o seu tempo de conclusão é apenas o tempo requerido para completar a própria tarefa. Neste cálculo, é suposto estarem disponíveis os recursos requeridos para realizar a tarefa.

Tarefa	Tempo para completar a tarefa	Dependências
A	8	
B	10	
C	8	A,B
D	9	A
E	5	B
F	3	C,D
G	2	D
H	4	F,G
I	3	E,F



**Diagrama PERT**

Tarefa	Tempo início	Tempo para completar a tarefa	Caminho crítico
<b>A</b>	0	8	
<b>B</b>	0	10	*
<b>C</b>	10	18	*
<b>D</b>	8	17	
<b>E</b>	10	15	
<b>F</b>	18	21	*
<b>G</b>	17	19	
<b>H</b>	21	25	*
<b>I</b>	21	24	

A partir do momento que os tempos para as tarefas a e b estão calculados, os tempos de conclusão para as tarefas c,d e e podem ser calculados. Já que os predecessores terminam em 8 e 10, as subtarefa c pode começar em 10 e completar-se em  $10+8=18$ . O tempo de início de d será 8 e o tempo de conclusão será  $8+9=17$ . Para e os tempo será  $10+5=15$ . Agora podemos calcular os tempos para as tarefas f e g. O tempo para  $f = 18+3 =21$  e para  $g=17+2=19$ . Os tempos de início de h e i é 21. O tempo de conclusão de  $h=21+4=25$  e  $i=21+3=24$ .

**Caminho crítico.** É o conjunto de tarefas que determina o menor tempo de conclusão possível. O tempo de conclusão será mais longo se não houverem recursos suficientes para realizar todas as tarefas paralelamente; entretanto, nunca pode ser encurtado adicionando mais recursos

#### **Algoritmo para marcar o caminho crítico**

1. Inicie com as tarefas com maior tempo de conclusão, marque-as como críticas
2. Seleccione os predecessores dessas tarefas com maior tempo de conclusão e marque-os como críticos
3. Continue o passo 2 até chegar aos nós iniciais

Na tabela podemos ver os tempos de conclusão de todas as tarefas. A tarefa h tem o melhor tempo de conclusão (25). Então, marcamos h como parte do caminho crítico. Os predecessores de h são f e g. A tarefa f tem o tempo de conclusão maior dessas duas tarefas, então f é marcado como parte do caminho crítico. A tarefa f tem como predecessores c e d. Já que c tem o tempo de conclusão maior, c é marcado comp parte do caminho crítico. A subtarefa c tem a e b como predecessores, e já que b tem o tempo maior, ele é parte do caminho crítico. Já que agora temos uma tarefa inicial, o caminho crítico está completo.

Folga. As subtarefas que não estão no caminho crítico possuem alguma flexibilidade para iniciarem. Essa flexibilidade é chamada folga.

Escolha a tarefa com o maior tempo de conclusão que não tenha sido processada. Se a tarefa não tem sucessor, escolha o maior tempo de término de todas as tarefas. Se a tarefa tem sucessor, escolha o mais recente dos maiores tempos de início dos nós sucessores. Esse é o maior tempo de conclusão para essa tarefa. Esse maior tempo de início para a tarefa reflecte o seu tempo

Repita o passo 2 até que todos caminhos não críticos tenham sido processados

**Exemplo:** A tarefa com maior tempo de conclusão é a tarefa i. Já que não tem sucessor, o maior tempo de conclusão será usado (25). Ele é adicionado como o maior tempo de conclusão para i. Já que é mais um dia do que o tempo de conclusão de i, a data de início é alterada de 21 para 21,22. Agora a ultima tarefa não crítica é g. Já que h é a única sucessora de g e h deve começar em 21, g deve terminar em 21. Então o tempo de conclusão g torna-se 19,21 e o tempo de início 17,19. A próxima tarefa não crítica a processar é d. Ele tem como sucessores f e g. A subtarefa f deve começar em 18, então a conclusão de d torna-se 17,18 e o início torna-se 8,9. Depois processa-se e que tem g e i como sucessores. g tem como último tempo de início 19 e i, 22, então o início de i torna-se 10,15 e a conclusão 15, 19. A ultima tarefa é a que tem como sucessores c e d. A tarefa a tem que ser completada em 9, então o tempo de conclusão será 8,9 e o tempo de início 9,1. Na tabela resume-se os resultados.

Tarefa	Tempo início	Tempo para completar a tarefa	Caminho crítico
A	0,1	8,9	
B	0	10	*
C	10	18	*
D	8,9	17,18	
E	10,14	15,19	
F	18	21	*
G	17,19	19,21	
H	21	25	*
I	21,22	24,25	

## PARTE II.

**Estimativas de custo do software.** A tarefa de estimativa de custo de software serve para determinar quantos recursos serão necessários para completar o projecto. Geralmente essa estimativa é feita em programador/meses. (PM) Existem várias abordagens diferentes para estimar os custos do software. A abordagem mais antigo é chamada estimativa LOC, já que ela é baseada em estimativas iniciais do número de linhas de código que será necessário para desenvolver o projecto. Outra abordagem é baseada na contagem de pontos de função na descrição do projecto. Mais recentemente, têm surgido as estimativas OO, baseadas em elementos dos diagramas UML. Por exemplo PM para desenvolver uma classe.

**Estimativa de LOC.** O primeiro passo das estimativa baseada em LOC é estimar o número de linhas de código no projecto final. Isso pode ser feito baseado em experiência, tamanho do projectos anteriores, tamanho da solução corrente ou dividindo-se o projecto

em partes menores e, então, estimando o tamanho de cada uma dessas partes. Uma abordagem padrão é, para cada parte, estimar o tamanho máximo possível, o tamanho mínimo possível e o melhor tamanho. A estimativa para o projecto todo é um sexto da soma dos tamanhos máximos, mínimos e 4 vezes o melhor tamanho.

**Exemplo.** A equipa WRT identificou as LOC das 7 partes no seu projecto.

Parte	Tamanho máximo	Melhor Tamanho	Tamanho Mínimo	Estimativa
1	20	30	50	$20+4*30+50=190/6=31,6$
2	10	15	25	$10+4*15+25=95/6=15,8$
3	25	30	45	$25+4*30+45=190/6=31,6$
4	30	35	40	$30+4*35+40=220/6=36,7$
5	15	20	25	$15+4*20+25=120/6=20$
6	10	12	14	$10+4*12+14=72/6=12$
7	20	22	25	$20+4*2+25=133/6=22,17$

A estimativa total para todo o projecto é  $31,6+15,8+31,6+36,7+20+12+22,17 = 170,07$  LOC.

**Estimativa de Custo baseada em LOC (COCOMO).** COCOMO é uma formula de estimativa de custo por LOC, criada por Barry Bohem nos anos 70. Ele usou como unidade milhares de instruções de código prontas como unidade de tamanho, KLOC é o equivalente. A sua unidade de esforço é o número de programador por mês (PM). Bohem dividiu os dados históricos do projecto em 3 tipos de projecto.

1. Programas aplicativos (destacado, orgânico, como gestão de dados ou processamento científico)
2. Programas utilitários (semi-destacados –“semi-detached”, como compiladores, linkeditores, analisadores)
3. Programas de sistemas (embebidos –embedded-)

Ele identificou os valores dos parâmetros para a determinação do esforço:

1. **Programas aplicativos:**  $PM = 2,4 * (KLOC)^{1,05}$
2. **Programas utilitários:**  $PM = 3,0 * (KLOC)^{1,12}$
3. **Programas de sistemas:**  $PM = 3,6 * (KLOC)^{1,20}$

Na tabela ilustra-se o esforço para projectos de 5 a 50 KLOC utilizando as fórmulas COCOMO

Tamanho	Apl	Util	Sist
5k	13,0	18,2	24,8
10k	26,9	39,5	57,1
15k	41,2	62,2	92,8
20k	55,8	86,0	131,1
25k	70,5	110,4	171,3

30k	85,3	135,3	213,2
35k	100,3	160,8	256,6
40k	115,4	186,8	301,1
45k	130,6	213,2	346,9
50k	145,9	239,9	393,6

Bohem também determinou que nos seus dados de projecto existia um tempo de desenvolvimento padrão baseado no tipo e no tamanho do projecto. As fórmulas a seguir são para calcular o tempo de desenvolvimento em programador/mês

1. **Programas aplicativos:**  $PM = 2,5 * (PM)^{0,38}$
2. **Programas utilitários:**  $PM = 2,5 * (PM)^{0,35}$
3. **Programas de sistemas:**  $PM = 2,5 * (PM)^{0,32}$

Na tabela ilustra-se o tempo de desenvolvimento para projectos de 5 a 50 KLOC utilizando as fórmulas COCOMO

Tamanho	Apl	Util	Sist
5k	6,63	6,90	6,99
10k	8,74	9,06	9,12
15k	10,27	10,62	10,66
20k	11,52	11,88	11,90
25k	12,60	12,97	12,96
30k	13,55	13,93	13,91
35k	14,40	14,80	14,75
40k	15,19	15,59	15,53
45k	15,92	16,33	16,25
50k	16,61	17,02	16,92

**Análise de ponto de função.** O objectivo dos pontos de função é identificar e quantificar as funcionalidades requeridas para um projecto. A ideia é contar coisas no comportamento externo que irão requerer processamento. Os itens clássicos de contagem são:

- Entradas
- Saídas
- Requisitos (Consultas)
- Arquivos internos
- Interfaces externas

*Requisitos ou consultas* são pares de solicitação resposta que não mudam os dados internos. Por exemplo, a solicitação pelo endereço dum funcionario específico é um requisito. Toda a sequência de perguntar, preencher o nome e o endereço seria contado como um requisito (consulta).

*Entradas* são itens da aplicação de dados suprida pelo programa. A entrada lógica geralmente é considerada um item e os campos individuais não são contados separadamente. Por exemplo, a entrada de dados pessoais para um funcionário pode ser considerada uma entrada.

**Saídas** são os dados da aplicação exibidos. Pode ser um relatório, uma tela ou uma mensagem de erro. Novamente, campos individuais não são considerados saídas separadas. Se um relatório tem múltiplas linhas, por exemplo, uma linha para cada trabalhador dum departamento, essas linhas seriam consideradas como uma saída. Entretanto, algumas autoridades contam linhas sumárias como saídas separadas.

Arquivos internos são arquivos lógicos que o utilizador entende e que devem ser mantidos pelo sistema. Se um arquivo actual contém 1.000 entradas de dados pessoais, esse seria provavelmente contado como um arquivo. Entretanto, se um arquivo contém dados pessoais, dados sumários do departamento e outros dados do departamento, eles serão contados como três arquivos separados para o propósito de contar os pontos de função.

**Interfaces externas** são dados que são compartilhados por outras programas. Por exemplo, um arquivo pessoal pode ser usado por recursos humanos para promoção ou para folha de pagamento. Então, ele pode ser considerado uma interface com ambos os sistemas.

**Contando os pontos de função não ajustados.** Os itens de pontos de função individuais são identificados e então classificados em simples, médio e complexo. Os pesos da tabela são então designados para cada tem e o total é somado. Esse total é chamado pontos de função não ajustados.

Não há um padrão para a contagem dos pontos de função e existem livros com regras de contagem. É importante lembrar que os pontos de função tentam medir a quantidade de esforço que será necessária para desenvolver um software. A tabela ilustra um exemplo de pesos atribuídos aos pontos de função não ajustados.

	<b>Simple</b>	<b>Médio</b>	<b>Complexo</b>
<b>Saídas</b>	4	5	7
<b>Consultas</b>	3	4	6
<b>Entradas</b>	3	4	6
<b>Arquivos</b>	7	10	15
<b>Interfaces</b>	5	7	10

**Exemplo:** O departamento quer um programa para distribuir horários e salas para cada secção e criar um calendário para os cursos. Há uma lista de secções com o nome do professor designado e o tamanho pré-definido, bem como uma lista das salas com a lotação máxima de cada sala. Existem ainda alguns conjuntos de aulas que não podem ser usados ao mesmo tempo. Além disso, professores não podem leccionar dois cursos ao mesmo tempo.

Este programa é muito mais difícil que a complexidade das entradas e saídas. Ele tem duas entradas principais: a do arquivo com a lista de secções, professores designados e tamanho pré-definido, e a do arquivo com a lista de salas com a lotação máima. Esses dois arquivos, embora simples para ler serão avaliados como complexos por serem difíceis de processar. Existirá um arquivo adicional com o conjunto de aulas que não podem ser ministradas ao mesmo tempo. Novamente, esse arquivo é simples em estrutura, mas difícil de processar. A última linha tem uma restrição que não é nenhuma entrada e nenhuma saída. Existe uma saída, o calendário, que é uma saída complexa. Não

há requisitos ou interfaces mencionadas, nenhuma menção sobre arquivos sendo mantidos.

**Produtividade.** Uma das medidas mais importantes é a produtividade dos desenvolvedores de software. Ela é determinada pela divisão do tamanho total do produto terminado pelo esforço total de todos os programadores, em unidades de LOC/programador/dia. Uma alternativa é medir a produtividade em termos de ponto de função por programador/dia. A produtividade inclui todos os esforços despendidos em todas as fases do ciclo de vida do software.

**Exemplo:** A companhia XYZ despendeu o esforço apresentado na tabela para cada fase do ciclo de vida no último projecto. Calcule o esforço em termos de LOC/programador/dia e em termos dos pontos de função/programador/dia. O ponto de função estimado foi 50 pontos de função não ajustados. O projecto terminado incluiu 950 LOC.

Fase	Programador/Dias
Requisitos	20
Projecto	10
Implementação	10
Teste	15
Documentação	10
TOTAL	65

O total de esforço foi 65 programadores/dia. Isto dá uma produtividade de  $950/65=14,6$  LOC/programador/dia. Usando pontos de função não ajustados, a produtividade é  $50\text{ pf}/65=0,77$  pf/programador/dias.

### Problemas

1. Crie uma WBS para a tarefa de pintar uma sala. Assuma que deve incluir três fases: plano de trabalho, compra de materiais, trabalho e limpeza
2. Crie uma WBS para o desenvolvimento de software dum produto para o seguinte consultório de dentista: “Quando um paciente telefonar para marcar uma consulta, a recepcionista irá verificar o calendário e oferecerá o primeiro horário disponível. Se o paciente concordar com a data e o horário propostos, a recepcionista entrará com a consulta, o nome do paciente e o horário proposto. O sistema irá verificar o nome e completar detalhes necessários a partir do arquivo do paciente, incluindo o bilhete de identidade. Após cada exame ou limpeza, o higienista ou assistente irá marcar a consulta como terminada, adicionando comentários e agendando nova visita, se necessário. O sistema irá executar buscas pelo nome do paciente e pela data. Detalhes dos arquivos do paciente são mostrados junto com as informações da consulta. A recepcionista pode tanto cancelar consultas como imprimir uma lista de notificação para lembrar de telefonar dois dias antes das consultas. O sistema inclui o número de telefone do paciente a partir do seu arquivo, ou ainda imprimir diaria ou semanalmente calendários de trabalho com todos os pacientes.”

3. Crie uma WBS para desenvolvimento de software dum produto para o seguinte problema dum pousada: “O Sandro e a Ana vão inaugurar uma pousada numa pequena vila do Alentejo. Eles terão três quartos para hóspedes e querem um sistema para gerenciar as reservas e para monitorar despesas e ganhos. Quando um cliente potencial telefonar para fazer a reserva, eles irão verificar a agenda e, se existir vaga, entrarão com o nome do cliente, endereço, número de telefone, datas, concordância do preço, número do cartão de crédito e número do(s) apartamento(s). As reservas devem ser garantidas pelo pagamento dum dia.. Reservas poderão ser feitas sem garantias até uma data definida. Se não confirmada até a data, a reserva será cancelada.
4. Desenhe um diagrama PERT para as tarefas do problema 1
5. Desenhe um diagrama PERT para o seguinte conjunto de tarefas e dependências. Completa a tabela mostrando o caminho crítico e as folgas

Tarefa	Dependência	Tempo	Início	Término
a		10		
b	a	5		
c	a	2		
d	a	3		
e	b,c	7		
f	b,d	9		
g	c,d	5		
h	e,f,g	6		

6. Desenhe um diagrama PERT para o seguinte conjunto de tarefas e dependências. Completa a tabela mostrando o caminho crítico e as folgas

Tarefa	Dependência	Tempo	Início	Término
a		10		
b	e	10		
c	d,f	10		
d	a,f,b	20		
e	a,f	8		
f	a	5		

7. Calcule o esforço COCOMO e o tempo de desenvolvimento, pessoal médio e produtividade para um projecto orgânico que é estimado em 39.800 LOC
8. Calcule os pontos de função não ajustados para o problema descrito no problema 2.