Avaliação do Portfolio de Sistemas Legados Usando Modelos de Estimação Algorítmicos

Fernando Brito e Abreu
INESC

Resumo

Este artigo inclui uma avaliação quantitativa do valor do portfolio de um conjunto de aplicações legadas numa grande empresa nacional. Esta avaliação é uma das peças necessárias à concepção de um modelo de custeio das operações de manutenção dessas aplicações, na fundamentação de acções de reengenharia ou mesmo na avaliação de serviços de desenvolvimento em "outsourcing". São utilizados dois modelos de estimação de recursos especialmente concebidos para o planeamento de projectos de Engenharia de Software e analisam-se as divergências encontradas.

1. Introdução

Desde os primórdios da computação que se iniciaram esforços para desenvolver modelos matemáticos para estimar os custos da produção de software. As primeiras tentativas consistiam em heurísticas do tipo:

"num grande projecto cada programador produz, em média, 150 linhas de código fonte testadas por mês".

"cada programador de manutenção mantém, em média, quatro caixas de cartões"

Só no final dos anos 70 e início dos 80, começaram a ser desenvolvidos modelos matemáticos que constituíram um salto qualitativo importante no estado da arte da área. Este tipo de modelos necessitam de ser afinados ("calibrados") para cada ambiente específico, por forma a representar a influência de vários factores que influenciam a produtividade. Isso traduz-se no aparecimento de "constantes" ou "parâmetros".

Para além do seu papel crucial no planeamento de novos projectos, estes modelos podem ser igualmente utilizados para estimar o valor de um sistema legado. Tal pode ser necessário seja para conceber uma justa repartição do esforço das operações de manutenção, para fundamentar acções de reengenharia por migração em vez de substituição, ou mesmo para avaliar sistemas desenvolvidos em "outsourcing".

Neste artigo serão utilizados dois modelos de estimação adequados para sistemas legados, o COCOMO (COnstructive COst Model) e os Pontos de Função (Function Points). Nas próximas secções será efectuada uma breve introdução a cada um destes modelos, simultaneamente com a sua instanciação para um caso de estudo. A finalizar são comparados e discutidos os resultados obtidos com cada um.

O caso de estudo refere-se à estimação dos recursos necessários para conceber um conjunto de aplicações legadas, produzidas numa grande empresa nacional. Por outras palavras, pretende-se estimar quanto esforço e quanto tempo foram necessários para criar essas aplicações, dado que, como habitualmente acontece, não foram efectuados registos dessas grandezas durante a sua concepção.

As aplicações legadas em causa estão escritas em COBOL com SQL embutido e, na versão estudada, tinham **74693 linhas de código fonte**, não considerando linhas de comentários e linhas em branco. A porção dessas aplicações relativa às interfaces (formulários e relatórios) produzidos com as ferramentas disponibilizadas pela ORACLE, não é considerada neste estudo.

2. O Modelo COCOMO

2.1 Introdução

Este modelo foi proposto por Barry Boehm [Boehm81, Boehm84], tendo sido construído e calibrado inicialmente a partir de informação de um número considerável de projectos concluídos. O autor afirma que a sua utilização tem permitido estimativas com um erro inferior a 20% em 70% dos projectos. O interesse em torno deste modelo levou à criação nos EUA de um "COCOMO Users' Group" que organiza reuniões desde 1985. Na Europa existe também um grupo de utilizadores, denominado EuroCOCOMO.

O COCOMO considerada três modos de desenvolvimento:

- **Modo orgânico** - aplicável a ambientes de desenvolvimento estáveis, com pouca inovação e a projectos com equipas de dimensão relativamente pequena;
- **Modo semi-destacado** - aplicável a projectos com características entre o modo orgânico e o embutido

- **Modo embutido** - aplicável no desenvolvimento de sistemas complexos embutidos em hardware (ex.: sistemas de tempo-real), com muita inovação, com restrições severas e/ou com requisitos muito voláteis.

O COCOMO considera ainda três "estágios" do modelo, à medida que vai sendo introduzido mais detalhe:

- **Modelo Básico**
- **Modelo Intermédio**
- **Modelo Completo**

2.2 Modelo Básico

Enquanto se vai apresentando o modelo vamos, simultaneamente, aplicando-o ao caso de estudo. Iremos considerar dois cenários: um pessimista (maior esforço, maior custo, calendário mais dilatado) e um optimista. A situação real estará situada, muito provavelmente, entre estes dois extremos. As aplicações em causa enquadram-se melhor no modo orgânico, que será o doravante utilizado.

2.2.1 Estimação do Esforço

O algoritmo básico para a estimação do esforço **E** a partir do número de linhas de código a desenvolver (L expresso em K) é

$$\mathbf{E = a\ (L)^{b}\ [homem.mês]}$$

em que as constantes **a** e **b** do modelo tomam os seguintes valores:

	a	**b**
Modo orgânico	2.4	1.05
Modo semi-destacado	3.0	1.12
Modo embutido	3.6	1.20

Estes valores são considerados típicos da indústria de software. Deverão porém ser sucessivamente refinados com base em projectos passados, por forma a obter um melhor ajustamento das curvas (calibração do modelo). Considera-se que um homem.mês corresponde em média a 152 horas de trabalho por cada mês de calendário. Este valor toma em consideração os dias de férias, formação e faltas por doença. Tem-se então que um homem.ano corresponde a 152 * 12 = 1824 homem.hora .

O esforço total previsto segundo o modelo básico é:

$$E = 2.4 * (74.693)^{1.05} = 222.4 \textit{ homem.mês} = 18.5 \textit{ homem.ano}$$

Note-se que este valor só toma em consideração a dimensão dos sistemas produzidos e não todas as outras facetas (humanas, tecnológicas e organizacionais) da complexidade que condicionaram a sua produção. Estas serão consideradas adiante quando utilizarmos o modelo ontermédio, pelo que a estimativa do esforço será substituída por outras consideradas mais credíveis.

2.2.2 Estimação do Prazo Nominal

Um calendário excessivamente dilatado provoca o efeito conhecido por *Lei de Parkinson* (expansão do trabalho até ao limite permitido), não promove bons hábitos de trabalho (é-se levado a sofisticar em excesso o sistema em desenvolvimento) e diminui a produtividade. Segundo B. Boehm, alargar o prazo a 150% do nominal, aumenta os custos de 110%. Por outro lado, um calendário excessivamente curto pode ser simplesmente impraticável. Há um limite para aquém do qual é impossível reduzir o prazo de desenvolvimento. Segundo B. Boehm, comprimir o prazo a 75% do nominal, aumenta os custos de 125%.

O modelo COCOMO considera assim existir um prazo de desenvolvimento óptimo, designado por **prazo nominal**, dado por :

$$\mathbf{T = 2.5\ (E)^{c}\ [meses]}$$

em que a constante **c** do modelo pode tomar um dos seguintes valores:

	c
Modo orgânico	0.38
Modo semi-destacado	0.35
Modo embutido	0.32

Tal como as anteriores constantes, também esta deverá ser sucessivamente refinada com base em projectos passados.

O Prazo Nominal previsto segundo o modelo básico é:

$$T = 2.5 * (222.4)^{0.38} = 19.5 \text{ meses}$$

Este prazo será adiante recalculado para o modelo intermédio (que inclui mais informação).

2.3 Modelo Intermédio

2.3.1 Determinação dos Factores Influenciadores do Custo

O Modelo Básico é necessariamente simplista, por considerar apenas um modelo baseado na dimensão e em três níveis discretos da complexidade dos sistemas a produzir. A fase seguinte de sofisticação do modelo, a que se chama Modelo Intermédio, corresponde a considerar a influência de um conjunto de vários factores, relativos quer ao sistema a produzir (produto) propriamente dito, quer ao suporte computacional (tecnologia utilizada), factor humano e organização do processo de desenvolvimento de software. A influência destes factores, em número de 15 no modelo originalmente proposto, deve ser avaliada numa escala discreta e ponderada, expressa na seguinte tabela:

Factores Influenciadores do Custo(C_i)	Acrónimo	Muito Baixa	Baixa	Média	Alta	Muito Alta	Extra Alta
RELATIVOS AO PRODUTO							
• Nível de fiabilidade requerida	RELY	0.75	0.88	1.00	1.15	1.40	
• Dimensão da base de dados	DATA		0.94	1.00	1.08	1.16	
• Complexidade do produto	CPLX	0.70	0.85	1.00	1.15	1.30	1.65
SUPORTE COMPUTACIONAL							
• Restrições ao tempo de execução	TIME			1.00	1.11	1.30	1.66
• Restrições ao espaço de armazenamento	STOR			1.00	1.06	1.21	1.56
• Volatilidade da máquina virtual	VIRT		0.87	1.00	1.15	1.30	
• Tempo de resposta do computador	TURN		0.87	1.00	1.07	1.15	
PESSOAL							
• Capacidade dos analistas	ACAP	1.46	1.19	1.00	0.86	0.71	
• Experiência no domínio da aplicação	AEXP	1.29	1.13	1.00	0.91	0.82	
• Capacidade dos programadores	PCAP	1.42	1.17	1.00	0.86	0.70	
• Experiência utilização da máquina virtual	VEXP	1.21	1.10	1.00	0.90		
• Experiência na linguagem de programação	LEXP	1.14	1.07	1.00	0.95		
PROCESSO							
• Adopção práticas de programação actuais	MODP	1.24	1.10	1.00	0.91	0.82	
• Uso de ferramentas actuais	TOOL	1.24	1.10	1.00	0.91	0.83	
• Cumprimento com prazo definido	SCED	1.23	1.08	1.00	1.04	1.10	

Para reduzir a subjectividade inerente à atribuição de um valor da escala aplicaram-se as heurísticas descritas em [Boehm84]. Para alguns dos factores é apenas indicado um valor da escala (a sombreado), enquanto que para outros são indicados dois distintos. Esta última situação corresponde a casos onde a dúvida subsistia na escolha do valor mais apropriado. Tal será utilizado para traçar dois cenários, um optimista e outro pessimista.

2.3.2 Estimação do Esforço

A equação de estimação do esforço para o Modelo Intermédio passa a ser:

$$E = a(L)^b \prod_{i=1}^{15} c_i$$

A constante **a**, quando utilizados estes 15 factores é corrigida para:

Modo orgânico	3.2
Modo semi-destacado	3.0
Modo embutido	2.8

Cenário Optimista: $E = 3.2 * (74.693)^{1.05} * 1.33 = 394.4$ *homem.mês* $= 32.9$ *homem.ano*

Cenário Pessimista: $E = 3.2 * (74.693)^{1.05} * 3.28 = 972.7$ *homem.mês* $= 81.1$ *homem.ano*

Partindo de um valor padrão de 8.000 contos por homem.ano (valor total dos encargos suportados pela empresa) temos:

Cenário optimista: *263.200 contos*

Cenário pessimista: *648.800 contos*

A este valor há que adicionar os relativos, entre outras coisas, a:

- custos de arrendamento ou amortização de instalações, equipamento e software (sistema, operativo, software de rede, ferramentas, SGBDs, ...) utilizados
- custos de manutenção de hardware e software
- formação
- transportes e comunicações

2.3.3 Estimação do Prazo Nominal

Com base na equação do prazo nominal já atrás utilizada temos:

$$\textit{Cenário Optimista:} \quad T = 2.5 * (394.4)^{0.38} = 24.2 \textit{ meses}$$

$$\textit{Cenário Pessimista:} \quad T = 2.5 * (972.7)^{0.38} = 34.1 \textit{ meses}$$

Note-se que a expansão do prazo é menor que a expansão do esforço. Com efeito o modelo COCOMO advoga que a produção de sistemas mais complexos envolva equipas mais numerosas, isto é, uma distribuição temporal de esforço mais intensa.

2.4 Modelo Completo

Na sua versão mais completa, o modelo COCOMO introduz facetas adicionais como a decomposição de um sistema de grande dimensão em subsistemas. Outras facetas correspondem à distribuição das estimativas de esforço e de prazo por fase e por actividade e à influência diferenciada de cada factor influenciador do custo por fase. Para mais detalhes sobre este modelo consulte-se [Abreu98c, Abreu98d]. Não vamos considerar, neste estudo, o modelo completo.

3. O Modelo de Pontos de Função ("Function Points Analysis")

3.1 Introdução

Inicialmente proposto por Alan Albrecht (então na IBM) [Albrecht81] é um dos modelos de estimação de maior aceitação, sobre o qual quer a comunidade académica, quer a industrial, têm reflectido regularmente [Behrens83, Knafl86, Dreger89, Low90, Jones91, Davis92, Betteridge92, Abreu94]. Inicialmente divulgado através dos grupos de utilizadores *Guide* e *Share*, foi posteriormente (1986) criado um "International Function Point Users' Group" (IFPUG), com sede em Westerville (Ohio, EUA), que publica normas e especificações sobre como efectuar contagens de pontos de função. Mais recentemente foi criado um "European Function Point Users' Group".
Este modelo permite estimar a dimensão / complexidade de um sistema a desenvolver a partir do número de características visíveis - funcionalidades - do ponto de vista do utilizador ou dos outros sistemas a que aquele esteja ligado. Tem assim a vantagem de poder ser aplicável numa fase inicial do ciclo de vida dos sistemas, a partir da especificação dos requisitos, desde que esta esteja suficientemente detalhada. Outra vantagem geralmente imputável a este modelo é a da independência face à metodologia de análise e à linguagem de programação.

Neste artigo será utilizado um refinamento dos Pontos de Função com o acrónimo de MKII [Symons88] que resultou do trabalho da Nolan, Norton & Co. (uma empresa de Tecnologias de Informação da Peat Marwick McLintock) no âmbito de um contrato com a Civil Aviation Authority do Reino Unido. As melhorias desta variante referem-se à adaptação das regras de contagem dos pontos de função, no sentido de incluir métodos e ferramentas actualmente utilizados (bases de dados relacionais, metodologias de análise estruturada, geradores de aplicações, etc.). Em [Symons91], para além da descrição das regras de contagem, descreve-se a calibração do modelo com base em dados de uma centena de projectos recentes e é descrita uma ferramenta denominada "Before You Leap", que suporta o seu cálculo com essa calibração. Para mais detalhes sobre este modelo consulte-se [Abreu98a, Abreu98b].

3.2 Determinação dos Pontos de Função Não Ajustados

Para o cálculo da dimensão são consideradas as seguintes características do sistema:

- Número de entradas externas (écrans de entrada ou alteração de dados);
- Número de saídas externas (écrans de saída ou listagens de dados);
- Número de interrogações (transacções de entrada/saída desencadeadas pelo utilizador);
- Número de ficheiros lógicos internos (acessíveis pelo utilizador).
- Número de interfaces externas (com outros sistemas, com outras aplicações, etc.)

Inicialmente é construída uma soma ponderada dos números acima referenciados. A ponderação é feita de acordo com critérios de complexidade (por exemplo o tipo e dimensão dos ficheiros de entrada e saída, o número de campos num écran, etc.), sendo a sua atribuição um exercício não trivial e moroso. A essa soma chama-se *Total Não Ajustado de Pontos de Função (TNAPF).*

	Simples	Médio	Complexo	Total
N.º entradas externas	___* 3 =	___* 4 =	___* 6 =	
N.º saídas externas	___* 4 =	___* 5 =	___* 7 =	
N.º interrogações	___* 3 =	___* 4 =	___* 6 =	
N.º ficheiros lógicos internos	___* 7 =	___* 10 =	___* 15 =	
N.º interfaces externas	___* 5 =	___* 7 =	___* 10 =	

TNAPF -»

No caso em apreço, devido a constrangimentos temporais, foi utilizado o método expedito a seguir descrito.

3.3 Cálculo Expedito de Pontos de Função

Alguns autores [Jones91, Davis92] afirmam existir uma proporcionalidade entre o total não ajustado de pontos de função (TNAPF) e o número de linhas de código fonte (LOC). Esta proporcionalidade dependeria da linguagem usada tal como consta na seguinte tabela:

Linguagem	**LOC / TNAPF**
Assembly	320
Macro Assembly	213
C	125 a 128
Cobol (ANSI)	91 a 105
Fortran, Algol, Jovial	100 a 105
Pascal	85 a 91
Modula-2	80
PL/1	75
Ada	71
Basic (ANSI)	64 a 90
LISP	64
APL	32
Objective-C	27
Smalltalk	21

Repare-se que esta abordagem expedita ao cálculo de Pontos de Função, conhecida por "*backfiring*", só é passível de aplicação para projectos já concluídos, em que se conhece, portanto, o número de linhas de código. Vamos ensaiar esta abordagem para o caso das aplicações legadas em estudo, comparando os resultados com os obtidos através do modelo COCOMO e daí extrair as possíveis conclusões.

Para o caso das aplicações consideradas, que na versão disponibilizada tem 74693 linhas de código em COBOL, tem-se que os Pontos de Função não ajustados deverão variar entre 74693/105 e 74693/91. Assim tem-se:

$TNAPF \in [711, 821]$

3.4 Ajustamento da Contagem

Seguidamente é calculado o factor de complexidade tecnológica (FCT), a partir da estimativa do grau de influência de 14 características do sistema em desenvolvimento. A tabela seguinte é preenchida com os valores da legenda para o caso das aplicações legadas.

	Optmista	Pessimista
comunicação de dados	3	4
funções distribuídas	2	3
desempenho	4	5
tipo de configuração	3	3
ritmo das transacções	4	5
entrada interactiva de dados	3	4
eficiência dos utilizadores	3	4
actualização interactiva	3	4
complexidade do processamento	3	3
reutilização	3	3
facilidade de instalação	2	3
facilidade de utilização	3	4
dispersão física das instalações	3	3
facilidade de alteração	3	3
GTI -»	42	51

Valores possíveis:

Não presente ou nenhuma influência = 0
Influência insignificante = 1
Influência moderada = 2
Influência média = 3
Influência significativa = 4
Influência forte = 5

O *Grau Total de Influência (GTI)*, é o somatório das várias influências. O *Factor de Complexidade Tecnológica (FCT)*, por sua vez, é dado por:

$$FCT = 0.65 + C * GTI$$

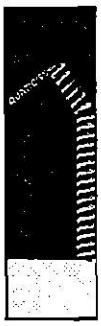
em que C, o *Coeficiente por Grau de Influência*, tem de ser calibrado com base em projectos passados. Como ponto de partida poderá usar-se o valor típico para a indústria (C = 0.01).

Para o caso em análise tem-se
$FCT \in [1.07, 1.16]$

Finalmente é calculado o *Total Ajustado de Pontos de Função (TAPF)* por multiplicação do total não ajustado pelo factor de complexidade tecnológica. Segundo este modelo TAPF é uma medida da dimensão S (size) do sistema considerado.

$$S = TAPF = TNAPF * FCT$$

$$S \in [761, 952]$$



3.5 Cálculo do Esforço e Produtividade

Seja:

- E o esforço [homem.hora]
- P a produtividade [pontos de função por homem.hora]
- P_L a produtividade em grandes projectos (TAPF > 1000)

Começa-se pelas equações da produtividade, que segundo o autor é constante para projectos acima de uma dada dimensão, isto é:

Para S > 1000

$P = P_L$

Na ausência de calibração pode usar-se o valor típico para a indústria (P_L = A * 0.06)

Para projectos de menor dimensão a produtividade tem um máximo para 250 pontos de função, decrescendo para um e outro lado desse ponto. A equação da produtividade é, segundo o autor do modelo, dada então por:

Para S < 1000

$$P = A * \left[0.11 * e^{-\left(\frac{S-250}{575}\right)^2} + \frac{0.01 * S^{1.1}}{522} \right]$$

em que A é um factor de escala a ser calibrado. Os valores típicos para a indústria são **A=1.0** (para linguagens 3GL) ou **A=1.6** (para linguagens 4GL).

Para o caso das aplicações legadas em que A = 1.0 e S ∈ [761, 952] obtém-se:

P ∈ [0.07824, 0.06099]

A partir da produtividade e da dimensão pode calcular-se o esforço:

$$P = S / E \qquad ou \qquad E = S / P$$

Tomando por base os factores de conversão considerados para o caso do modelo COCOMO:

- 1 homem.mês = **152** homem.hora
- 1 homem.ano = **12** homem.mês = **1824** homem.hora

tem-se:

E ∈ [9726, 15610] homem.hora = [63.99, 102.69] homem.mês = [5.33, 8.56] homem.ano

3.6 Determinação do Calendário

Seja

- T o período de tempo necessário para construir um sistema [semanas]
- D o ritmo médio (*delivery rate*) a que um sistema é construído [pontos de função por semana]
- K_D uma constante de calibração (valor típico para a indústria = 0.45)

$$D = K_D * \sqrt{S}$$

$$T = \frac{S}{D} = \frac{S}{K_D * \sqrt{S}} = \frac{\sqrt{S}}{K_D}$$

Para o caso das aplicações legadas tem-se S ∈ [761, 952], de onde se obtém finalmente:

D ∈ [12.41 , 13.88]

T ∈ [61.30 , 68.57] semanas = [14.15 , 15.82] meses.

4. Conclusões

A tabela seguinte resume os resultados obtidos com os dois modelos:

Modelo	*Prazo Optimista (meses)*	*Prazo Pessimista (meses)*	*Esforço Optimista (homem.ano)*	*Esforço Pessimista (homem.ano)*
COCOMO	*24.2*	*34.1*	*32.9*	*81.1*
P. de F. MkII	*14.2*	*15.8*	*5.3*	*8.6*

Como se pode verificar, existe uma acentuada discrepância nas estimativas, muito particularmente nas do esforço. Seria porém expectável que o modelo de Pontos de Função permitisse a obtenção de estimativas próximas das obtidas com o modelo COCOMO as quais foram, como já se disse, validadas. Vamos seguidamente levantar e discutir três hipóteses para corrigir esta aparente anomalia.

Hipótese 1 - Os valores típicos para a indústria assumidos para as constantes C, A e K_D são aceitáveis, mas o factor de conversão de Linhas de Código para Pontos de Função proposto em [Jones91] e [Davis92] está incorrecto.

- **Fundamento teórico**: o factor de conversão atrás utilizado pressupunha a utilização de linguagem COBOL apenas. Contudo, no caso em apreço, uma percentagem relativamente grande das linhas de código fonte do sistema em análise, estão expressas em SQL. Esta última linguagem, eminentemente declarativa, tem um poder expressivo superior ao COBOL, pelo que com o mesmo número de linhas, permite obter um maior funcionalidade, logo um maior número de Pontos de Função. Por outras palavras, o factor de conversão deverá ser menor.

- **Ensaio**: através do método de aproximações sucessivas foram determinados valores do factor de conversão que permitem obter intervalos de estimativa que estão contidos nos calculados com base no modelo COCOMO. Obtiveram-se os seguintes valores:

Cenário	*LOC/FP*	*FP*	*Esforço (FP)*	*Esforço (COCOMO)*	*Prazo (FP)*	*Prazo (COCOMO)*
Optimista	22	3633	33.2	32.9	30.9	24.2
Pessimista	20	4332	39.6	81.1	33.8	34.1

- **Discussão**: como se pode observar na tabela precedente conseguiu-se uma sobreposição de ambos os intervalos (esforço e calendário), o que não nos permite, à priori, refutar esta hipótese; contudo isso foi conseguido à custa de um considerável aumento do número de Pontos de Função (inicialmente tinha-se FP $\in$ [*761, 952*]) obtido à custa de uma grande redução do factor de conversão (inicialmente tinha-se LOC/FP $\in$ [91, 105]); esta redução, um tanto drástica, faz com que esta hipótese não se afigure inteiramente satisfatória.

Hipótese 2 - O factor de conversão proposto em [Jones91] e [Davis92] é aceitável, mas as constantes C, A e K_D tem de ser calibradas.

- **Fundamento teórico**: a constante "C" (Coeficiente por Grau de Influência) afecta o Factor de Complexidade Tecnológica que, ao influenciar a contagem de Pontos de Função, vai ter impacto nas duas estimativas (Esforço e Calendário); a constante "A" afecta a Produtividade, logo a estimativa do Esforço; a constante "K_D" (*delivery rate*) afecta a estimativa do Calendário. Todas estas constantes devem ser ajustadas com base em projectos passados. A esta operação de ajuste se chama **calibração** do modelo.

- **Ensaio**: através do método de aproximações sucessivas foram determinados valores das três constantes que permitiram obter intervalos de estimativa que estão contidos nos calculados com base no modelo COCOMO. Obtiveram-se assim os seguintes valores:

Cenário	***C***	FCT	*FP*	***A***	*Esforço (FP)*	*Esforço (COCOMO)*	K_D	*Prazo (FP)*	*Prazo (COCOMO)*
Optimista	0.02	1.49	1060	0.29	33.4	32.9	0.3	25.0	24.2
Pessimista	0.02	1.67	1371	0.29	43.4	81.1	0.3	28.5	34.1

- **Discussão**: embora se tenha conseguido uma sobreposição de ambos os intervalos (esforço e calendário), o que não nos permite, *à priori*, refutar esta hipótese, os valores das constantes que permitem este ajustamento são consideravelmente distintos (em particular A que é relativa à produtividade) dos valores típicos da indústria sugeridos na literatura, pelo que esta hipótese também não se afigura inteiramente satisfatória.

Hipótese 3 - Quer as constantes C, A e K_D, quer o factor de conversão de Linhas de Código para Pontos de Função têm de ser calibrados.

- **Fundamento teórico**: reunião dos fundamentos alegados para as duas hipóteses precedentes.

- **Ensaio**: através do método de aproximações sucessivas foram determinados valores do factor de conversão e das três constantes de calibração que permitem obter intervalos de estimativa que estão contidos nos calculados com base no modelo COCOMO. Obtiveram-se os seguintes valores:

Cenário	*LOC/FP*	***C***	FCT	*FP*	***A***	*Esforço (FP)*	*Esforço (COCOMO)*	K_D	*Prazo (FP)*	*Prazo (COCOMO)*
Optimista	61	0.02	1.49	1824	0.5	33.3	32.9	0.4	24.6	24.2
Pessimista	51	0.02	1.67	2446	0.5	44.7	81.1	0.4	28.5	34.1

- **Discussão**: as constantes, se bem que calibradas, são mais próximas dos valores típicos da indústria que no caso da hipótese 2. O factor de conversão é menor que o sugerido para COBOL,

dada a utilização de SQL embutido. Esta é uma hipótese de compromisso, provavelmente retractando melhor a situação real.

Note-se que em qualquer uma das hipóteses, a estimativa pessimista do esforço fornecida pelo modelo COCOMO é sempre bastante superior (cerca do dobro) às obtidas por via do modelo de Pontos de Função. Qualquer tentativa para obter valores semelhantes daquela estimativa através deste último modelo obrigavam à utilização de constantes de calibração e/ou factores de conversão com valores decididamente irrealistas. A explicação para tal facto poderá residir em algum exagero na determinação dos Factores Influenciadores do Custo no cenário (excessivamente) pessimista traçado com base no modelo COCOMO.

Para efectuar um estudo de validação mais exacto é necessário calcular o total de Pontos de Função através da forma convencional (referida na secção 3.2). Esse resultado permitirá uma melhor discussão da equivalência e grau de precisão destes dois modelos.

Bibliografia

[Abreu98d] Abreu, Fernando Brito : "Do ADA COCOMO ao COCOMO 2", InterFace, n°7, Maio de 1998.

[Abreu98c] Abreu, Fernando Brito : "Modelo COCOMO: das origens à actualidade", InterFace, n°6, Abril de 1998.

[Abreu98b] Abreu, Fernando Brito : "Pontos de Função: pistas e variantes", InterFace, n°5, Março de 1998.

[Abreu98a] Abreu, Fernando Brito : "Pontos de Função: uma história de sucesso?", InterFace, n°4, Fevereiro de 1998.

[Abreu94] Abreu, Fernando Brito & Jesus, Lurdes Pereira : "Experience with the Function Points MkII approach within software development efforts that use Relational Databases", actas da ESCOM Conference, Ivrea (Itália), Maio de 1994.

[Albrecht81] Albrecht, Allan J. : "Function Points as a Measure of Productivity", Actas do *53rd meeting of GUIDE International Corp.*, Dallas, 1981.

[Behrens83] Behrens, C.A. : "Measuring the Productivity of Computer Systems Development Activities with Function Points", IEEE Transactions on Software Engineering, Vol. 9, n° 6, p. 648-652, Novembro 1983.

[Betteridge92] Betteridge, R. : "Successful Experience of Using Function Points to Estimate Project Costs Early in the Life-Cycle", Information and Software Technology, Vol.34, n°10, p.655-658, Outubro 1992.

[Boehm81] Boehm, Barry W. : "Software Engineering Economics", Prentice-Hall Inc., Englewood Cliffs, NJ, 1981.

[Boehm84] Boehm, Barry W. : "Software Engineering Economics", IEEE Transactions on Software Engineering, Volume 10, N°1 , p. 4-21, Janeiro 1984.

[Davis92] Davis, Dwight B. : "Develop Applications On Time, Every Time", Datamation, p.85-89, Novembro 1, 1992.

[Dreger89] Dreger, J.Brian : "Function Point Analysis", Prentice-Hall, Englewood Cliffs, NJ, ISBN 0-13-332321-8, 1989.

[Jones91] Jones, T. Capers : "Applied Software Measurement: Assuring Productivity and Quality", McGraw-Hill (New York), ISBN 0-07-032813-7, 1991.

[Knafl86] Knafl, G.J. & Sacks, J. : "Software Development Effort Prediction Based on Function Points", Actas da *COMPSAC 86*, p.319-325, IEEE Computer Press, Outubro 1986.

[Low90] Low, Graham C. & Jeffery, D.Ross : "Function Points in the Estimation and Evaluation of the Software Process", IEEE Transactions on Software Engineering, Vol16, n°1, Janeiro 1990.

[Symons88] Symons, Charles R. : "Function point analysis: difficulties and improvements", IEEE Transactions on Software Engineering, Vol.14, N°1, p.2-11, Janeiro 1988.

[Symons91] Symons, Charles R. : "Software Sizing and Estimating - Mk II FPA (Function Point Analysis)", John Wiley & Sons, ISBN 0-471-92985-9, 1991.

Palavras-chave: Engenharia de Software, Estimação, COCOMO, Pontos de Função.

Temas da conferência abrangidos:

- Modelos de estimação de recursos em projectos de desenvolvimento
- Métricas de complexidade do software
- Avaliação dos serviços de "Outsourcing"
- Considerações económicas na gestão da qualidade