Publicado originalmente nas Actas das 6as Jornadas para a Qualidade no Software, APQ, Lisboa, Dezembro de 1992.

As Métricas na Gestão de Projectos de Desenvolvimento de Sistemas de Informação

Fernando Brito e Abreu (INESC/ISEG)
INESC - Instituto de Engenharia de Sistemas e Computadores
Rua Alves Redol, nº9, 1000 Lisboa
tel: +351-1-3100226
fax: +351-1-525843
e-mail: fba@si2000.inesc.pt

Resumo
A actividade do gestor de um projecto de desenvolvimento de sistemas de informação, à semelhança do que sucede noutras áreas, distribui-se pelas fases de planeamento, organização, selecção de pessoal, direcção e controlo. Um dos instrumentos indispensáveis para o exercício dessas funções, em todas as fases referidas, é o conhecimento de dados quantitativos sobre os sistemas em desenvolvimento ou a desenvolver e sobre os próprios processos de desenvolvimento. Essa informação quantitativa é expressa através daquilo que é habitual designar de "métricas de software" e que corresponde a uma área de investigação com grande interesse, cujo impacto se prevê importante, a julgar pelas experiências em outros países.
Este artigo pretende dar uma panorâmica das métricas mais conhecidas e seu enquadramento, das razões para a sua utilização, da implementação de um plano de métricas e dar a conhecer várias iniciativas em prol da disseminação da sua utilização. Dado que a temática das métricas está dispersa por muitas e variadas publicações, faz-se referência a uma alargada bibliografia como forma de facilitar futuras pesquisas.

Palavras-chave*: Desenvolvimento de Software; Métricas de Software; Programas de Métricas; Projectos ESPRIT; Gestão de Projectos de Engenharia de Software.*

1. INTRODUÇÃO

> *" ... when you can measure what you are speaking about, and express it in numbers, you know something about it; when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science ... "*
>
> *Lord Kelvin*

A manterem-se as tendências actuais, em algum instante futuro, não muito longínquo, a maioria da população activa estará envolvida directa ou indirectamente nos processos de desenvolvimento e manutenção de software[1]. Contudo o panorama actual desta área é caracterizado por custos e prazos incorrectamente estimados, má qualidade dos produtos finais e por uma taxa de crescimento global da produtividade que é inferior à da procura, razão pela qual é habitual falar-se da "crise de software". O Department of Trade and Industry do Reino Unido, por exemplo, estima que no seu país se desperdicem anualmente cerca de 500 milhões de libras (aproximadamente 110 milhões de contos) devido à má qualidade do software produzido. Afirma ainda que esses custos se situam em cerca de 20% do volume de negócios para uma empresa dedicada ao desenvolvimento de software, podendo 25 a 50% dessas perdas ser recuperadas através da implementação de um sistema de gestão da qualidade [TickIT92]. Assim sendo, para uma empresa com um volume de facturação de 200 mil contos a poupança situar-se-ia nos 10 a 20 mil contos anuais.

A produtividade depende de inúmeros factores como a dimensão e complexidade dos sistemas a desenvolver, as linguagens de programação, o grau de reutilização ou a experiência e motivação dos intervenientes no processo de desenvolvimento. Para constatar uma variação de produtividade resultante da influência desses factores é necessário medir e quantificar a produtividade.

---

1 - Estima-se que já em 1990 cerca de metade da população activa norte americana dependia de computadores e respectivo software para a execução do seu trabalho diário [Mills88].

A qualidade dos produtos de software é traduzida através de características como a correcção, eficiência, fiabilidade, portabilidade ou facilidade de manutenção. A obtenção de dados quantitativos relativos a essas características é assim fundamental para introduzir melhorias no processo de desenvolvimento.

A medição deve fazer assim parte do processo de produção de software, tal como é corrente na generalidade dos processos industriais, no sentido de permitir a previsão e monitorização dos custos, controlar a qualidade e, por outro lado, melhorar a compreensão e validação do próprio processo de desenvolvimento. Sem a obtenção desses dados quantitativos torna-se difícil, senão impossível, estabelecer, de uma forma racional, condições contratuais (prazos e orçamentos), tomar decisões de atribuição de recursos humanos e financeiros, ou tentar atingir objectivos definidos. A esta área do conhecimento convencionou-se chamar *métricas de software*.

A investigação e a aplicação dos seus resultados na área das métricas tem sofrido um acréscimo acentuado nos últimos anos, em particular na Europa, traduzida por exemplo por vários projectos ESPRIT. Outro aspecto sintomático dessa crescente importância é o aparecimento de conferências, "workshops", seminários e ferramentas dedicados a esta temática.

Das características mensuráveis dos produtos e processos de software, somente as que respeitam à avaliação do desempenho (tempos de resposta, capacidade de processamento, ...) não são geralmente incluídas na temática das métricas de software. Com efeito existe uma área de investigação, com um certo grau de autonomia, que trata precisamente das questões relacionadas com a avaliação do desempenho [Ferrari86].

2. PORQUÊ UTILIZAR MÉTRICAS ?

Embora as métricas não sejam a panaceia para todos os males no processo de desenvolvimento de software, quando utilizadas em conjunção com outras actividades, tais como um recrutamento cuidadoso dos elementos das equipas de desenvolvimento e de controlo de qualidade, formação adequada e atempada, a sua utilização tem sido apontada como um factor catalisador da qualidade e produtividade.

Os resultados habitualmente referidos na aplicação de um programa[2] de métricas são, entre outros:

- a melhoria da compreensão da actividade de gestão, dado que a obtenção de dados quantitativos faz desvanecer alguma da "imprevisibilidade" associada ao desenvolvimento de software;
- um controlo mais efectivo da qualidade, especialmente importante na fase de pré-instalação nos clientes finais;
- identificação mais precoce de problemas, permitindo acções correctivas em tempo útil, com menor impacto nos prazos globais dos projectos;
- melhoria da imagem da entidade, dado que a confiança dos clientes aumenta ao sentirem uma preocupação (e acção) no sentido de melhorar a qualidade dos produtos e serviços fornecidos;
- maior produtividade;
- identificação das fases do ciclo de vida, definidas no processo de desenvolvimento adoptado, onde um maior investimento em melhorias provocará uma maior rentabilidade;
- melhoria das estimativas e planeamento dos projectos;
- avaliação da eficácia (e eficiência) da introdução de novas ferramentas;
- avaliação da influência da linguagem e ambiente utilizado na produtividade e qualidade dos sistemas desenvolvidos;
- mudança positiva de atitute da parte dos intervenientes no processo de desenvolvimento, motivada pelo facto de se estarem a medir parâmetros do produto e do processo que possam realmente provar que se está a evoluir positivamente. O eventual alcance de objectivos expressos quantitativamente torna mais fácil a motivação dos intervenientes no processo de desenvolvimento através de apelos ao ego;
- maior facilidade dos gestores de projectos na comprovação, perante a direcção, da melhoria da proficiência conseguida;

Embora todos os benefícios acima referenciados constituam uma inegável vantagem competitiva para as organizações que lancem programas de métricas, muitas fazem-no apenas no intuito de cumprirem com os requisitos necessários para a sua

[2] - entendido como um conjunto integrado de actividades

certificação [Linkman91]. A este propósito transcreve-se um trecho de [ISO9000-3] integrado na sua secção 6 - "Sistema de Qualidade - Actividades de Suporte", dedicado às métricas:

*"**6.4.1 Medição do produto***

Deverão ser divulgadas e utilizadas métricas relevantes do produto de software em particular, para gerir os processos de desenvolvimento e de entrega (aos clientes finais).

Actualmente não existem medidas da qualidade do software universalmente aceites. Contudo deverão ser utilizadas, pelo menos, algumas métricas que traduzam as falhas e/ou defeitos do ponto de vista do cliente. As métricas seleccionadas deverão ser descritas de forma a que os resultados sejam comparáveis.

O produtor de produtos de software deverá recolher e agir com base nas medidas de qualidade desses produtos. Essas medidas deverão ser utilizadas com as seguintes finalidades:

a) para recolher dados e divulgar valores das métricas numa base regular;

b) para identificar o nível de desempenho baseado em cada métrica;

c) para levar uma acção correctiva se os níveis da métrica piorarem ou se excederem os níveis de objectivo definidos;

d) para estabelecer objectivos de melhoria em termos das métricas.

6.4.2 Medição do processo

O fornecedor deverá obter medidas quantitativas da qualidade dos processos de desenvolvimento e de entrega. Essas métricas deverão reflectir:

a) quão bem está a ser levado a cabo o processo de desenvolvimento em termos de cumprimento atempado de marcos e objectivos;

b) quão efectivo é o processo de desenvolvimento na redução das probabilidades de inserção de defeitos no software e de não detecção de defeitos existentes.

Tal como nas métricas de produto, o que interessa é que sejam utilizadas no controlo e melhoria do processo de desenvolvimento e não quais as métricas específicas que são utilizadas. A escolha das métricas deverá enquadrar-se no processo adoptado e, se possível, ter um impacto directo na qualidade do software produzido. Para diferentes produtos do mesmo produtor poderão ser apropriadas diferentes métricas."

3. DEFINIÇÃO E TAXONOMIA

Em termos formais, uma métrica é um indicador ou escala que permite quantificar um atributo de um produto do processo de desenvolvimento de software ou do próprio processo de concepção. O termo produto tanto se poderá aplicar a definições de requisitos, como a especificações funcionais, código fonte ou código executável.

Uma "boa" métrica deverá auxiliar no desenvolvimento de modelos capazes de estimar os parâmetros de um processo ou produto e não apenas descrevê-los. Para tal deverá ter as seguintes características:

- Mensurabilidade, para o que deve ter uma definição inequívoca, por forma a ser claro como é determinada;
- Repetibilidade, isto é, tomar o mesmo valor, mesmo quando calculada por pessoas diferentes, ou em instantes distintos;
- Baixo custo, para o que deve ser fácil e rápida de obter (de preferência de uma forma automática);
- Validade, ou seja, permitir medir, com objectividade, aquilo para que foi proposta;

- Robustez, isto é, não ser muito sensível a variações não significativas do processo ou produto.

Como adiante se constatará, poucas, senão mesmo nenhumas métricas possuem estes atributos na sua totalidade.

São vários os possíveis enquadramentos classificativos das métricas. Serão a seguir mencionados três deles: o *objecto* das métricas, isto é, o âmbito da sua aplicação, o *critério* utilizado na sua determinação e o *método* de obtenção.

Quanto ao *objecto*, as métricas podem ser globalmente classificadas em duas categorias:

- as **métricas de produto** são aplicáveis aos produtos de qualquer das fases de desenvolvimento, quantificando, por exemplo, a sua dimensão, complexidade ou qualidade (fiabilidade, facilidade de utilização, manutenção ou outras propriedades);

- as **métricas de processo** referem-se ao processo de concepção e desenvolvimento e podem referir-se a grandezas tais como a duração do desenvolvimento, custos totais ou o nível de experiência dos programadores.

Quanto ao *critério*, podem ser definidas igualmente duas categorias:

- as **métricas objectivas**, são geralmente obtidas através de regras objectivas e bem definidas, única forma de possibilitar comparações posteriores consistentes. Na prática isso significa que os valores a que se chega deveriam ser sempre os mesmos, independentemente do instante, condições ou indivíduo que os determina. A obtenção destas métricas é passível de automatização;

- as **métricas subjectivas** são baseadas em atributos cuja medição não pode ser feita senão de uma forma subjectiva, sendo muitas vezes derivadas de resultados de questionários e entrevistas. A complexidade, por exemplo, é por vezes classificada numa escala de "simples" / "abaixo da média" / "média" / "acima da média" / "muito grande"[3]. Embora a generalidade dos projectos possa ser facilmente classificada nestas categorias, aqueles que se situam perto das fronteiras podem ser "encaixados" na estrutura de classificação de formas distintas, dependendo da subjectividade do técnico medidor. Um outro exemplo é a quantificação da experiência de um programador.

Quanto ao *método* de obtenção, podem ser classificadas como:

- métricas **elementares** ou **primitivas** se são a expressão de um atributo único, como por exemplo a duração do período de validação ou o número de ecrans de entrada e saída de dados;

- métricas **compostas** ou **computadas** se são calculadas com base em outras métricas, como por exemplo o número de falhas por milhar de linhas de código.

4. AS MÉTRICAS NO PROCESSO DE TOMADA DE DECISÃO

A análise diacrónica, em que se verifica se os indicadores macroscópicos estão, ou não, a evoluir significativamente para os níveis pretendidos, é a abordagem mais habitual. Esses indicadores são geralmente computados a partir das métricas para o total dos projectos concluídos e em curso. Alguns desses indicadores, podem ser calculados com base na média, variância ou desvio padrão de métricas.

A comparação inter-projecto poderá permitir aquilatar das vantagens e inconvenientes entre diferentes metodologias utilizadas, diferentes ambientes de desenvolvimento ou diferentes procedimentos de gestão. Se, por exemplo, os valores normalizados das métricas de um dado projecto se desviarem significativamente (inclusivé no sentido positivo) dos valores típicos para a organização, então esse projecto deverá ser alvo de uma auditoria cuidada no sentido de identificar quais os factores que motivaram tais desvios. Com esse conhecimento poder-se-á evitar a repetição de erros (ou adoptar procedimentos e técnicas de êxito assim comprovado).

[3] - Em [ISO9126] propõe-se a escala: excelente / bom / razoável / pobre.

O critério para a selecção dos projectos que necessitam de ser inspecionados poderá ser o de considerar apenas aqueles que estiverem na região exterior a um intervalo em torno dos valores típicos. A amplitude desse intervalo deverá ser reduzida sempre que se verifique que apenas uma diminuta parte dos projectos apresentem valores fora desse intervalo. Pode adoptar-se como regra, por exemplo, que a amplitude do intervalo é reduzida de 5% sempre que menos de 25% dos projectos apresentem valores fora do mesmo. Outra regra possível seria a de inspecionar quando o desvio fosse superior ao desvio padrão.

Os projectos "não conformes" poderão ter associado, por exemplo, uma incorrecta ou não actualizada avaliação da complexidade que afecte toda a normalização, uma desadaptação dos recursos humanos envolvidos, ou uma má especificação ou interpretação dos requisitos. Embora os desvios por excesso sejam mais habituais, poderão igualmente ocorrer desvios por defeito. Se, por exemplo, a fase de análise de determinado projecto envolveu um esforço normalizado consideravelmente inferior à média, tal poderá denotar uma abordagem demasiado redutora que terá certamente consequências graves por não ter atingido o nível de detalhe que o utilizador final irá exigir. Por outras palavras, nem sempre um valor abaixo dos níveis típicos será um indicador de progresso no sentido desejado.

Considere-se, por exemplo, que se traça a evolução temporal do nº comulativo de pedidos de alterações, em oposição às alterações que foram realmente efectuadas (e as correspondentes versões instaladas nos clientes). É de esperar que as duas curvas sejam sensivelmente paralelas e mesmo convergentes. Se houver lugar a divergência, isso será um sinal inequívoco de problemas subjacentes.

Muitas vezes só se conseguem tirar conclusões quando várias métricas são analisadas conjuntamente. Considere-se, por exemplo, dois projectos X e Y com complexidades e dimensões semelhantes. Contudo, enquanto que no projecto X a produtividade foi de 10 linhas de código por pessoa por dia, no projecto Y foi de 50. As conclusões poderiam ser precipitadas se não se considerasse que no projecto X a taxa de falhas foi de 0,1 por 1000 de linhas código produzidas, enquanto esse valor para o projecto Y foi de 15 !

Entenda-se portanto que a análise das métricas é uma valiosa ajuda para a detecção de potenciais disfunções e não um meio de diagnóstico (nem de cura) rigoroso e definitivo. Elas podem ser usadas no sentido de monitorizar o progresso alcançado, permitindo a identificação de potenciais problemas por forma a desencadear uma acção correctiva em tempo útil. Cada caso em que esse mecanismo de alerta for accionado deverá ser investigado, no sentido de detectar se existe realmente um problema ou se o "aviso" é espúrio.

A análise de métricas de projectos concluídos permite ajudar no estabelecimento de condições contratuais (prazos e orçamentos) e na tomada de decisões de atribuição de recursos humanos, técnicos e financeiros. As métricas de dimensão, por exemplo, têm sido utilizadas na estimação da *produtividade* dos programadores [Banker89, Yu90], logo permitindo estimar um calendário para o desenvolvimento. A tentação de utilizar métricas para, num futuro mais ou menos próximo, avaliar o desempenho dos intervenientes no processo de desenvolvimento, tem contudo de ser bem ponderada. Com efeito o conhecimento actual sobre as métricas ainda não está consistentemente estruturado nesta área, pelo que a sua utilização prematura poderia conduzir a conclusões descabidas. Note-se também que o uso de métricas para esses fins poderá provocar distorsões futuras dos dados subjacentes.

Têm sido feitos vários estudos em que algumas métricas são correlacionadas com características do software como sejam a *fiabilidade*, *complexidade*, *facilidade de manutenção* ou o *custo total de desenvolvimento*. Essas correlações, quando "afinadas" para um dado ambiente, podem ser e têm já sido [Li87, Rombach87] utilizadas para efectuar previsões e a partir destas tomar as decisões de gestão consentâneas.

O período de depuração ("debugging") é uma das fases do processo de controlo de qualidade de um sistemas que mais poderá beneficiar da utilização de métricas [Brown89, Takahashi89, Gorla90]. Com efeito, é habitual verificar-se que os módulos em que são detectados mais erros durante o desenvolvimento, são também aqueles que são mais sujeitos a erros depois da entrega do sistema aos utilizadores finais. A obtenção sucessiva de medidas da frequência, incidência e da razão subjacente às alterações correctivas de um sistema (nas especificações, inclusivé), além de identificar áreas potenciais de problema ou de má compreensão, permite garantir que o processo de teste proceda eficientemente a um ritmo aceitável.

As métricas, contudo, nem sempre têm sido utilizadas de uma forma sistemática e metódica. Com efeito o estado da arte nesta área caracteriza-se por uma certa profusão de métricas propostas, mas em que poucas têm sido validadas. Mesmo no caso das métricas mais estudadas, como a LOC, Halstead e McCabe, não há um consenso generalizado sobre aquilo que

pretendem quantificar. Quando são utilizadas, os resultados em ambientes diversificados nem sempre são corroborantes, o que, acrescendo à falta de fundamentação teórica da maioria das métricas, levou a que apenas algumas tenham aceitação generalizada[4]. A subjectividade de algumas das métricas, como o grau de competência / experiência de um programador ou o grau de utilizabilidade de um produto a desenvolver, também não favorece a sua utilização, embora se aguardem avanços na sua definição mais objectiva num futuro próximo, como resultado da investigação e experimentação em curso.

5. EXEMPLOS DE MÉTRICAS

5.1 MÉTRICAS DE DIMENSÃO E COMPLEXIDADE

Têm sido propostas inúmeras métricas para medir a dimensão e complexidade de um programa. Optou-se por apresentá-las em conjunto pois, a maior parte das vezes, a mesma métrica é apresentada como quantificando ora a dimensão, ora a complexidade, consoante os autores.

O número conhecido de métricas deste tipo é grande e não pára de aumentar. Em [Li87], por exemplo, são referidas e comparadas 31 métricas de complexidade diferentes. Em [Card88, Harrison87] são propostas novas métricas de complexidade. Seguidamente referem-se algumas das mais mencionadas e supostamente utilizadas.

Linhas de Código Fonte (LOC - "Lines Of Code")

É uma métrica de dimensão que, apesar de criticada, é ainda a mais utilizada [Mills88]. Embora aparentemente simples, obriga a uma definição inequívoca das regras de contagem de linhas, nomeadamente no tocante ao tratamento a dar às linhas em branco e de comentário, instruções não executáveis, directivas de compilação, múltiplas instruções por linha ou múltiplas linhas por instrução, bem como no caso de reutilização de código.

Em [Jones91] é descrito um conjunto de possíveis regras para a contagem de linhas[5]. Para além da definição do método de cálculo, só podem ser comparados valores diferentes desta métrica se se referirem à mesma linguagem e se o estilo de programação estiver normalizado.

Numerosos estudos como [Curtis79a, Curtis79b, Woodfield81] têm tentado validar correlações entre as LOC e a complexidade de um programa, os esforços de desenvolvimento e de manutenção, a produtividade de um programador[6], bem como com outras métricas sobre o mesmo programa.

A deficiência mais óbvia, apontada a esta métrica, é a de que só pode ser determinada quando o processo de codificação termina.

Métricas de Halstead

É um conjunto de métricas proposto por Maurice Halstead [Halstead77], que se baseia na teoria da informação[7] e que o autor designou por "Software Science". O volume informativo de um módulo de um programa (medido em bits...) em termos do seu comprimento e do seu vocabulário, que corresponde ao número de "tokens" (operadores e operandos) num programa, é dado por:

$$\text{Volume} = \text{Comprimento} * \log_2 \text{Vocabulário}$$

---

4 - Este facto é evidenciado em [ISO9126].

5 - A regra de cálculo mais universal é a de que todas as linhas que não sejam, nem de comentário, nem em branco, sejam consideradas, independentemente do número de instruções por linha.

6 - Imagine-se, contudo, qual será a atitude de um programador que descobre que está a ser avaliado pelo número de linhas produzido ! Além disso, nem sempre interessa produzir muito código. Veja-se o caso dos sistemas embebidos em que tipicamente se pretende o contrário...

7 - É reconhecido como sendo o primeiro conjunto de métricas com uma base teórica.

onde

Comprimento = somatório das instâncias de (operadores + operandos) utilizados

Vocabulário = somatório (operadores + operandos) diferentes utilizados

A determinação deste volume não é difícil pois não envolve qualquer análise semântica. Note-se que o somatório dos volumes de vários módulos calculados separadamente, não é igual ao volume calculado de uma só vez sobre a concatenação dos mesmos módulos. Contudo há análises críticas [Lassez81] à abrangência desta métrica, em particular na suposta independência face à linguagem de programação utilizada.

Em [Levitin86] defende-se que a métrica de comprimento de Halstead é objectiva e melhor que a LOC. Em [Shen83] faz-se uma avaliação crítica desta métrica. Em [Christensen81, Curtis79b] relatam-se duas experiências bem sucedidas da sua aplicação. Em [Li87, Christensen81] apresentam-se evidências empíricas de que as métricas de comprimento e de volume de Halstead estão linearmente relacionadas com a LOC.

Métricas de McCabe

A primeira e mais conhecida métrica proposta por Thomas McCabe [McCabe76] é a *métrica de complexidade ciclomática*. Esta pressupõe que a complexidade depende do número de decisões, é adimensional e corresponde ao número máximo de percursos linearmente independentes através de um programa. Estes podem ser representados através de um grafo orientado em que os nós representam uma ou mais instruções sequenciais e os arcos orientados indicam o sentido do fluxo de controlo entre várias instruções. Numa linguagem estruturada, isso equivale ao número total de pontos de decisão (embutidos nas estruturas de controlo de iteração e selecção), mais dois.

Em [Gaffney79] apresenta-se um estudo em que se conclui que a produtividade diminui de forma não linear, à medida que aumenta a densidade de pontos de decisão. Outros estudos correlacionam também esta métrica com os esforços de depuração e manutenção. A métrica ciclomática pode ser então utilizada para estimar custos a partir do desenho detalhado dos módulos. Em [Harrison82, Woodfield81, Curtis79b] são relatadas experiências de utilização desta métrica. Alguns investigadores proposeram alterações no cálculo desta métrica no sentido de melhorar a sua validade [Myers77, Stetter84]. Em [Henry81a] reporta-se a determinação de um elevado grau de correlação entre esta métrica e a de Halstead. A objectividade e o interesse em torno desta métrica levou inclusivamente à sua normalização [NBS82].

Posteriormente McCabe definiu outras cinco métricas: *complexidade da unidade real, complexidade essencial, complexidade do desenho do módulo, complexidade do desenho total e complexidade da integração*. Estas métricas têm sido utilizadas [Mannino92] para identificar e minimizar código não estruturado, decidir sobre o número de testes necessários para uma cobertura total das hipóteses de execução, eliminar lógica redundante e para restringir a complexidade de módulos produzidos a um nível aceitável.

Métrica dos Nós

É uma métrica proposta por Woodward [Woodward79] que corresponde ao número de cruzamentos (nós) no desenho que traduz a sequência de controlo de execução de instruções num programa.

Métrica dos Fluxos de Informação

Foi proposta por Henry e Kafura [Henry81b, Kafura81] no pressuposto de que os fluxos de informação na estrutura de um programa[8] (interconectividade) são uma medida da sua complexidade.

A métrica é calculada para quantificar a complexidade de cada procedimento ou módulo, da seguinte forma

[8] - Daí que por vezes seja utilizada a denominação de **métricas de estrutura.**

complexidade = comprimento $*$ (fan_in $*$ fan_out)^2

em que o "fan_in" é o número de parâmetros de entrada do módulo e "fan_out" o número de parâmetros de saída.

Em [Henry84] reporta-se uma experiência de validação desta métrica, no âmbito de um projecto de concepção e desenvolvimento de um núcleo para o sistema operativo UNIX. Em [Kafura85, Kafura87] são relatadas experiências de utilização combinada de várias métricas (incluindo esta). Em [Rombach87] refere-se que esta métrica permite estimar com uma precisão considerável o esforço de manutenção.

Métrica de Oviedo

É uma métrica de complexidade que é calculada a partir da soma ponderada de uma métrica de controlo de fluxo (número mínimo de percursos independentes) e uma métrica de utilização de dados ("data usage metric") [Rook90].

Dado ser uma métrica complexa, não se torna claro perceber qual é o atributo que se pretende medir, o que dificulta a sua interpretação e utilização.

Índice de Nevoeiro de Gunning

É uma métrica de complexidade aplicada sobre a documentação, a qual, para ser utilizável, deverá ser expressa de uma forma compreensível. É calculada a partir do número de palavras por frase e do número de palavras tri-silábicas utilizadas [Rook90]. A sua validade é obviamente discutível.

Métrica Bang

É uma métrica proposta por Tom DeMarco [DeMarco82], que pretende medir a dimensão total de um sistema desenvolvido e que é determinada a partir das funcionalidades descritas nas especificações formais. O algoritmo de cálculo é diferente consoante se aplique a sistemas orientados por dados ou orientados por processos. O seu autor propõe que um dos objectivos de um gestor de projecto seja o de maximizar o quociente Bang / Custo total. Outros investigadores têm proposto métricas sobre a complexidade do desenho de sistemas [Troy81, Henry84, Yau85, Card88].

5.2 MÉTRICAS DE QUALIDADE

É longa a lista de características geralmente associadas à qualidade do software, como a correcção, eficiência, portabilidade, fiabilidade, facilidade de utilização (utilizabilidade), modularidade, grau de reutilização ou facilidade de manutenção. Contudo, algumas destas características sobrepõe-se ou são mesmo antagónicas, como por exemplo a eficiência e a portabilidade (quando uma aumenta a outra diminui), o que obstou a que fosse possível encontrar uma métrica que quantificasse a qualidade globalmente.

Para além dos exemplos a seguir referidos, têm sido propostas métricas de qualidade noutras áreas, embora com menor enfâse. Em [Myers75, Yin78], por exemplo, propõem-se métricas para a qualidade na fase de desenho.

5.2.1 Métricas de Utilizabilidade

Um dos esforços conhecidos nesta área cujo produto final se designa SUMI[9] (Software Usability Measurement Inventory) é reportado em [Kirakowski92] onde a utilizabilidade é definida como "a capacidade segundo a qual um produto pode ser utilizado com eficiência e satisfação para atingir objectivos específicos em ambientes específicos".

[9] - O SUMI foi derivado de investigação realizada no âmbito dos projectos Esprit HUFIT (nº385) e MUSIC (nº5429).

A utilizabilidade é, nesta aproximação, desdobrada em 5 vertentes, cada uma com uma ponderação própria:

- Facilidade de aprendizagem - capacidade de um utilizador atingir rapidamente um grau de proficiência elevado;
- Controlo - capacidade de um produto para responder de uma forma natural e consistente aos comandos e entradas de dados fornecidos;
- Utilidade - capacidade de resolver ou ajudar a resolver problemas para o qual o produto foi proposto;
- Eficiência - capacidade de resolver problemas de forma rápida e económica;
- Afecto - capacidade de um produto interagir amigavelmente com os seus utilizadores;

Um dos problemas deste tipo de métricas é que as escalas não têm sentido em valor absoluto. Só podem extrair-se conclusões quando se comparam dois ou mais produtos. Para além disso têm de existir versões executáveis dos produtos e vários utilizadores que conheçam consistentemente as suas potencialidades.

5.2.2 Métricas de Manutenção Correctiva

Para os utilizadores finais as falhas encontradas e a sua eficaz e eficiente eliminação são sempre o ponto mais sensível na avaliação que fazem da qualidade dos produtos que utilizam e dos processos da entidade produtora. Note-se que a noção de falha abarca também a não conformidade com a análise dos requisitos. Nesta área têm sido propostas métricas como:

- o número de falhas detectadas por inspecção ao código
- o número de falhas detectadas por aplicação de uma bateria de testes
- o número de falhas detectadas pelos utilizadores[10]
- o tempo médio de resposta na resolução de problemas detectados
- o número de alterações efectuadas no código fonte

Tomando o enquadramento classificativo proposto na secção 3, as métricas acima referidas são *objectivas*, de *produto* e *compostas*.

5.2.3 Métricas de Fiabilidade

Tem sido feita investigação no sentido de obter métricas de estimar o número de defeitos existentes num programa e assim dimensionar o esforço de depuração (por exemplo a quantidade de testes a aplicar). Partindo das métricas de manutenção correctiva o propósito é assim o de quantificar a *fiabilidade* do software. Métricas típicas deste tipo são a probabilidade de uma falha ocorrer num intervalo de tempo especificado (MTTF - "Mean Time To Fail"), ou o tempo médio entre falhas (MTBF - "Medium Time Between Failures").

Têm sido efectuados vários estudos em que se tenta relacionar a fiabilidade com métricas de dimensão e complexidade, como as de Halstead e de McCabe [Potier82, Shen85].

Os trabalhos mais relevantes nesta área tentam relacionar as métricas de complexidade com a questão da manutenção [Curtis79a, Harrison82, Rombach87]. A complexidade é aí medida quer em termos internos, de cada módulo, quer em termos das interrelações entre módulos. A correlação entre a complexidade e a manutenção pode ser assim explorada no sentido de conseguir a tão desejada redução dos custos de manutenção, através de recomendações a seguir no desenho

[10] - estes números de falhas aparecem por vezes normalizados pela dimensão do sistema a que se referem.

detalhado dos sistemas [Yau80, Yau85, Kafura87]. Outras referências de interesse nesta temática são [Musa75, Ruston79, Musa80, Musa87, Brown89, Takahashi89, Gorla90].

5.2.4 Métricas de Reutilização

Esta é uma área emergente onde ainda são conhecidas poucas propostas ou resultados. A disponibilização (e o crescente interesse em torno) de metodologias, técnicas e ferramentas para o desenvolvimento baseado no paradigma da orientação por objectos, vem dar uma ênfase particular à questão da reutilização. A métrica mais óbvia é a percentagem de código reutilizado que incorpora um produto final. No caso de desenvolvimento utilizando um SGBDOO, poderá por exemplo adoptar-se o número de classes reutilizadas sobre o número total de classes.

6. ESTABELECIMENTO DE UM PROGRAMA DE MÉTRICAS

A implementação de um plano de métricas numa organização que desenvolva software inclui habitualmente [Abreu92, Perez92, Pyramid91] os seguintes passos:

a) avaliação do ambiente de desenvolvimento dos projectos por forma a determinar o nível de maturidade actual e a permitir uma definição consistente dos objectivos para o plano.

Para estabelecer o ponto de partida que servirá de base para a quantificação da evolução desejada, torna-se necessário lançar uma acção de auditoria ao estado da arte da prática actual, alargada a todos os projectos em curso. Tais acções deverão ser efectuadas por consultores independentes, como forma de não "viciar os dados" à partida. Em [AMI91] propõe-se que o enquadramento metodológico mais apropriado para estas acções de auditoria seja o do "Capability Maturity Model" [Paulk91, Paulk92a, Paulk92b] proposto pelo Software Engineering Institute (SEI) sediado na Universidade de Carnegie-Mellon [11]. Este modelo permite uma classificação segundo uma escala de 5 níveis segundo critérios bem definidos, o que permite divisar acções a empreender para caminhar no sentido da optimização. Um esforço de auditoria semelhante baseado no ISO 9000 seria bastante mais problemático dado não haver no âmbito deste standard uma estrutura quantitativa de classificação que permita inequivocamente avaliar os diversos estágios de evolução.

b) decomposição hierárquica dos objectivos em sub-objectivos e identificação de métricas a partir destes últimos.

Note-se que sem uma abordagem sistematizada existe o perigo de medir:
- de menos, logo não obtendo o conhecimento necessário para tomar medidas;
- demais, logo ficando "afogado" num mar de números de difícil correlação e compreensão;
- as grandezas erradas, logo desvirtuando as conclusões finais.

O conjunto de métricas seleccionado não deverá ser estático, isto é, algumas das métricas inicialmente propostas poderão ser abandonadas e outras poderão ser adoptadas em sua substituição.

c) definição das responsabilidades, calendários e procedimentos e obtenção de ferramentas de apoio à recolha e tratamento dos dados;

d) recolha e validação dos dados sobre os produtos e processos do seu desenvolvimento.

O processo de obtenção de métricas deve ser uma actividade independente, tanto quanto possível, das do projecto alvo. Com efeito, é geralmente aplicável uma versão ligeiramente adaptada do "Princípio de Incerteza de Heisenberg" [DeMarco82]:

"Medir um parâmetro de um projecto e atribuir-lhe um significado visível, afecta a utilidade dessa medição."

Por outras palavras, se fôr conhecido, por parte dos intervenientes no desenvolvimento, que um dado parâmetro (as linhas de código fonte, por exemplo) está a ser utilizado para a quantificação da produtividade individual, pode pensar-se nos efeitos

[11] - Pittsburg, EUA.

contraproducentes que seriam gerados. A solução para esse problema, poderá passar pela constituição de uma equipa de recolha de métricas independente da equipa de desenvolvimento. Por outro lado deverá ser recolhido um conjunto de métricas suficientemente diversificado, por forma a minimizar a probabilidade de uma viciação.

e) análise das métricas computadas com base nos dados recolhidos e sua confrontação com os objectivos definidos para auxiliar a tomada de decisão no sentido de melhorar quer o processo de desenvolvimento, quer o próprio plano de métricas.

A aplicação de um programa de métricas não dá resultados imediatos[12], embora acarrete um esforço de implementação que tem obviamente custos imediatos. Tal facto tem desanimado alguns gestores de projectos que têm abandonado os programas em curso. Outras razões habitualmente apontadas como causadoras do insucesso na implementação de planos de métricas são a adopção de métricas irrelevantes ou mal compreendidas, acções erradas derivadas de má interpretação das métricas, excessivos custos de recolha, falta de apoio dos elementos da gestão ou pura e simplesmente a não utilização das métricas na tomada de decisão. Tais situações são infelizmente mais frequentes do que seria de esperar, de tal forma que os detractores das métricas chegam mesmo a afirmar que a maior parte dos esforços conhecidos sobrevivem apenas 1 a 2 anos [Meredith92].

Reveste-se assim de enorme valor a promoção e sensibilização de todos os intervenientes no processo no sentido de "vender" este tipo de iniciativas. Tais acções podem traduzir-se na organização de seminários internos, grupos de trabalho, distribuição de "papers" relevantes, etc. Para conseguir um maior cometimento deve dar-se ênfase a aspectos como a capacidade de diagnosticar potenciais problemas mais cedo, a melhoria do processo de planeamento, a possibilidade de reconhecer software bem feito e a inovação no contexto nacional que tal esforço representa. Por outro lado as pessoas gostam, em geral, de comparar o seu desempenho com o das normas, se estas forem exequíveis e especialmente se participarem na sua definição [Peters82]. Deve também ser feita publicidade a histórias de sucesso externas (no início) e internas (à medida que sejam observadas melhorias). Assim acrescenta-se mais um passo, que deverá desenrolar-se continuadamente desde o início da implementação

f) promoção das vantagens esperadas e motivação de todos os intervenientes para a participação geral na implementação do plano de métricas.

7. AS FERRAMENTAS

Dado que o cálculo manual de métricas é um processo moroso e repetitivo, a aceitação da sua utilização passa, para além da motivações já referidas, pela disponibilização de ferramentas que automatizem esse processo. A maioria das ferramentas disponíveis comercialmente operam sobre o código fonte, logo são dependentes da linguagem de programação utilizada. Possuem potencialidades que podem agrupar-se em duas categorias, a da análise estática e a da análise dinâmica.

A análise estática envolve as fases típicas de um compilador (análises léxica e sintática) e destina-se, em geral, a estimar características da qualidade como a modularidade e a portabilidade ou a complexidade. Os resultados da análise são em geral traduzidos não só em métricas mas frequentemente também através de representações gráficas.

A análise dinâmica só pode ser efectuada se houver uma versão executável, pois que o analisador é em geral "linkado" ao software a analisar. Os objectivos da análise dinâmica, efectuada tipicamente na fase de teste final dos sistemas, são os de obter um melhor conhecimento sobre o seu comportamento em situações extremas (fiabilidade), o cumprimento dos requisitos no tocante à vertente temporal (eficiência) ou mesmo a completitude da funcionalidade pretentida.

Em [Jannasch92] é apresentado o resultado de um levantamento efectuado recentemente na Europa e que inclui as seguintes ferramentas: ATHENA (Athens Technology Center - Grécia), LOGISCOPE (Verilog - França), METROPOL (CEP Systemes / Matra / INRIA - França), QUALIGRAPH (SzKI Computer Research and Innovation Center - Hungria), QUALMS (CSSE / South Bank Polytechnic - Reino Unido) e TESTSCOPE (Scope GmbH - Alemanha). O perfil típico das ferramentas aí descritas é o seguinte:

[12] - Em [Pyramid91] reporta-se que o período normal para a obtenção de melhorias consistentes na qualidade e na produtividade é habitualmente de dois anos.

• suportam a análise de programas escritos em C, ADA, Fortran, Cobol, Pascal e (menos frequentemente) C++, CHILL, PLM e Modula 2;

• possuem capacidades da análise estática (gráficos de chamadas de funções e procedimentos, gráficos com a estrutura de controlo, ...) e dinâmica ("branch coverage", testes de validação tipo "caixa-preta", ...);

• computam várias métricas como a Complexidade Hierárquica e Estrutural (Gilb), Complexidade Ciclomática (McCabe), Nível de Profundidade do Aninhamento (Dunsmore / Gannon), Medida da Testabilidade (Mohanty), Densidade de Controlo (Szentes / Portiers), Medidas da Ciência de Software (Halstead);

• possuem geralmente um sistema de armazenamento persistente dos resultados das análises efectuadas como forma de facilitar posteriores comparações, refinamentos e tomadas de decisão;

• estão disponíveis em ambiente UNIX e com interface gráfica X-Windows e mais raramente em MSDOS e VMS;

• produzem resultados na forma textual, tabelar ou gráfica (diagramas de Kiviat, histogramas, etc).

Como se pode constatar, a totalidade das ferramentas mencionadas utilizam como entrada os programas fonte e por isso permitem computar métricas objectivas de produto (segundo o enquadramento proposto na secção 3).

8. O ESFORÇO COMUNITÁRIO E AS EXPERIÊNCIAS CONHECIDAS

No plano europeu, foram lançados nos últimos anos, no âmbito do programa ESPRIT da Comissão das Comunidades Europeias, vários projectos visando a investigação aplicada e a divulgação das métricas [ESPRIT90]. São eles:

MUSE - "Software Quality and Reliability Metrics for Selected Domains"
(Proj. nº1257 - Janeiro 87 a Dezembro 89)

O objectivo deste projecto foi o de produzir métricas de qualidade e fiabilidade, demonstráveis no âmbito de vários domínios aplicacionais selecionados. O consórcio, liderado pela Brameur (Reino Unido), incluia empresas da França, Alemanha e Grécia.

METKIT - "Metrics Education Toolkit"
(Proj. nº 2384 - Fevereiro 1989 a Janeiro 1992)

Os objectivos principais deste projecto eram os de aumentar a divulgação e fomentar a utilização de métricas de software na indústria europeia, através da produção de material educacional dirigido quer à comunidade industrial, quer à académica [Bush91, Fenton91]. O consórcio incluia instituições do Reino Unido, Alemanha, Dinamarca, França e Itália e era liderado pela Brameur Ltd (Reino Unido).

PYRAMID - "Promotion of Metrics"
(Proj. nº 5425 - Janeiro 1991 a Dezembro 1992)

O fito principal deste projecto é o de ajudar a melhorar a qualidade e produtividade nas actividades de desenvolvimento e manutenção de software na Europa, através da utilização de métodos quantitativos. Os seus promotores esperam que os resultados concorram para uma utilização crescente de métricas de software, reduzindo o atrazo existente face aos EUA e Japão [Pyramid91]. O "prime contractor" do PYRAMID é a Veridatas (França), reunindo o consórcio outras instituições do Reino Unido, Alemanha, Grécia, Itália e França.

MUSIC - "Metrics for Usability Standards in Computing"
(Proj. nº 5429 - Novembro 1990 a Novembro 1993)

O objectivo deste projecto é o desenvolvimento de uma metodologia modular que possa ser utilizada para especificar os requisitos mínimos para a utilizabilidade de um produto e para certificar se este produto cumpre com os requisitos especificados. Nesse sentido estão a ser desenvolvidos um modelo de referência, métricas, procedimentos e um curso de

formação. Os participantes pretendem também disseminar os conceitos e o método MUSIC em comités de normalização e na comunidade industrial europeia das T.I. em geral [Kirakowski92]. O consórcio é formado por universidades e outras instituições do Reino Unido, Itália, Espanha, Alemanha, Irlanda e Holanda, sendo liderado pela Brameur Ltd (Reino Unido).

AMI - "Application of Metrics in Industry"
(Proj. nº 5494)

O objectivo principal do projecto AMI é o de demonstrar a maturidade e a aplicabilidade da tecnologia das métricas à comunidade da indústria de software. Como produto principal os promotores indicam o "Metrics User's Handbook" [AMI91] que é um guia para a progressiva adaptação, instalação e aplicação do modelo de métricas proposto pelo AMI, o qual esperam venha a tornar-se num standard "de facto" no seio da Comunidade [Perez92]. O consórcio, coordenado pela GEC Marconi Ltd (Reino Unido), integra várias instituições da França, Austria, Alemanha, Espanha, Itália e Reino Unido.

Note-se que instituições de praticamente todos os países comunitários, entre "software-houses", empresas de consultadoria, universidades e institutos de investigação, têm participado nestes projectos, primando as portuguesas por uma total ausência.

Para além dos projectos trans-europeus acima mencionados, foram já organizadas duas conferências dedicadas ao tema das métricas, EUROMETRICS 91 (Paris) e EUROMETRICS 92 (Bruxelas), onde também não houve participação nacional. Outras conferências como a "European Conference on Software Quality" (Oslo 1990, Madrid 1992) incluíram uma percentagem significativa de comunicações e "tutorials" relacionados com a temática dos métodos quantitativos aplicados à Engenharia de Software.

Quanto a experiências bem sucedidas, existem variadas referências, um pouco por todo o mundo [Linkman91, Pyramid91, Grady87], como por exemplo nas seguintes organizações:

- Siemens Nixdorf Informationsysteme AG - SNI (Munique - Alemanha), Rheinisch-Westfälischer TÜV (Alemanha), Bull AG (Koeln - Alemanha), GEC-Marconi Software Systems (Borehamwood - Reino Unido), Data Logic (Harrow - Reino Unido), CRIL (Rennes - França), Corelis Technologie (França), GEC-Alsthom (França), ITS - Ingenieria y Tecnologia de Sistemas (Espanha), Advanced Software Technology SRL (Itália), Alcatel-ELIN (Austria), Hewlett Packard [13], Siemens Medical Electronics (Danvers - EUA), Texas Instruments (EUA), Hitachi (Japão) e NEC (Japão).

9. CONCLUSÕES

Para combater a crise de software são necessárias melhores estimativas de custos e prazos, melhor qualidade e maior produtividade, o que só pode ser conseguido com uma gestão mais eficaz do processo de desenvolvimento de software. Contudo, essa gestão é extremamente dificultada devido à complexidade envolvida e à escassez de medidas bem definidas e fiáveis que permitam guiar e avaliar, quer os produtos, quer o próprio processo da sua concepção e desenvolvimento. A melhoria da actividade de gestão nesta área passa pela capacidade de identificar, medir e controlar os principais parâmetros que afectam o processo de desenvolvimento de software, bem como aqueles que são intrínsecos aos produtos (dimensão, complexidade ou qualidade), além das suas interdependências. Esse é precisamente o âmbito das métricas de software.

Há bastante tempo que têm vindo a ser propostas métricas e modelos nelas baseados. Alguns estudos detalhados [Grady87, Pyramid91, AMI91] indicam que a implementação de um programa de métricas ajuda na obtenção de melhores resultados do ponto de vista da gestão, quer no curto prazo (para um dado projecto) quer no longo prazo (projectos futuros) melhorando a estimação, a produtividade e a qualidade. Com efeito, um número cada vez maior de organizações tem vindo a obter resultados promissores com a implementação de programas de métricas. Estes indicadores positivos têm alertado a comunidade de gestores de projectos, pelo que se espera que, a médio prazo, também nas empresas com projectos de menor dimensão (como ocorre no tecido empresarial português desta área) se possam vir a colher idênticos benefícios.

Urge dominar e utilizar métodos quantitativos para melhorar a qualidade e produtividade do software "made in Portugal". No INESC está a desenvolver-se, em colaboração com o ISEG, uma área de investigação aplicada em métricas e modelos de estimação. Espera-se que o "know-how" daí resultante venha a ser oportunamente transferido para a comunidade empresarial.

[13] - Em 25 locais de I&D espalhados pelos EUA, Europa, Japão e Austrália.

Referências

[Abreu92] Abreu, Fernando Brito : "Um Plano de Métricas para o Centro de Sistemas Computacionais", relatório técnico, INESC/ISEG, 1992.

[Albrecht81] Albrecht, Allan J. : "Function Points as a Measure of Productivity", Actas do *53rd meeting of GUIDE International Corp.*, Guidance for users of integrated data processing equipment conference, Dallas, 1981.

[AMI91] AMI (Application of Metrics in Industry) Consortium : "Metric Users' Handbook - A Quantitative Approach to Software Management", ESPRIT Project 5494 (AMI), 1991.

[Banker89] Banker, Rajiv D. & Kemerer,Chris F. : "Scale Economies in New Software Development", IEEE Transactions on Software Engineering, Vol. 15, nº 10, p. 1199-1205, Outubro 1989.

[Behrens83] Behrens, C.A. : "Measuring the Productivity of Computer Systems Development Activities with Function Points", IEEE Transactions on Software Engineering, Vol. 9, nº 6, p. 648-652, Novembro 1983.

[Bento92] Bento, João Nuno & Carapuça, Rogério : "Organização do Processo de Desenvolvimento de Software", actas das 6ªS Jornadas de Qualidade no Software, Associação Portuguesa para a Qualidade (APQ), Dezembro 1992.

[Brown89] Brown, David B., Maghsoodloo, Saeed & Deason, William H. : "A Cost Model for Determining the Optimal Number of Software Test Cases", IEEE Transactions on Software Engineering, Vol. 15, Nº2, p.218-221, Fevereiro 1989.

[Bush91] Bush, M. & Russel, M. : "A New Modular Course For Teaching about Software Engineering Measurement", Actas da Conferência ESPRIT´91, Comissão das Comunidades Europeias, DGXIII, 1991.

[Card88] Card, David N. & Agresti, W.W. : "Measuring Software Design Complexity", Journal of Systems and Software, Vol. 8, nº3, p. 185-197, 1988.

[Christensen81] Christensen, K. & Fitsos, G. P. & Smith, C. P. : "A Perspective on Software Science", IBM Systems Journal, Vol. 20, nº 4, p. 372-388, Outubro 1981.

[Curtis79a] Curtis, B. & Sheppard, S.B. & Milliman, P. & Borst, M.A. & Love, T. : "Managing the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics", IEEE Transactions on Software Engineering, Vol. 5, nº2, p. 96-104, Março 1979.

[Curtis79b] Curtis, B. & Sheppard, S.B. & Milliman, P. : "Third Time Charm: Stronger Prediction of Programmer Performance by Software Complexity Metrics", Actas da *4th International Conference on Software Engineering*, IEEE, p. 356-360, Setembro 1979.

[DeMarco82] DeMarco, Tom : "Controlling Software Projects: Management, Measurement and Estimation", Prentice-Hall, Englewood Cliffs, NJ ou Yourdon Press, New York, 1982.

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

[ESPRIT90] European Strategic Programme for Research and Development in Information Technology : "Synopses of Information Processing Systems - ESPRIT Projects", DGXIII, Comissão das Comunidades Europeias, Setembro 1990.

[Fenton91] Fenton, N. E. (editor) : "Software Metrics: A Rigorous Approach", Chapman & Hall (UK) ou Van Nostrand Reinhold (USA), ISBN 0-412-40440-0, 1991.

[Ferrari86] Ferrari, D. : "Considerations on the Insularity of Performance Evaluation", IEEE Transactions on Software Engineering, Vol. 12, nº6, p. 678-683, Junho 1986.

[Gaffney79] Gaffney, J.E. : "Program Control Complexity and Productivity", in [IEEE79], p. 142, 1979.

[Gorla90] Gorla, Narasimhaiah, Benander, Alan C. & Benander, Barbara A. : "Debugging Effort Estimation Using Software Metrics", IEEE Transactions on Software Engineering, Vol. 16, Nº2, p.223-231, Fevereiro 1990.

[Grady87] Grady, Robert B. & Caswell, Deborah L. : "Software Metrics - Establishing a Company-Wide Program", Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Halstead77] Halstead, Maurice H. : "Elements of Software Science", Elsevier North-Holland, New York, 1977.

[Harrison82] Harrison, W. & Magel, K. & Kluczny, R. & DeKock, A. : "Applying Software Complexity Metrics to Program Maintenance", Computer, Vol. 15, nº9, p. 65-79, Setembro 1982.

[Harrison87] Harrison, W. & Cook, C. : "A Micro/Macro Measure of Software Complexity", Journal of Systems and Software, Vol. 7, nº3, p. 213-219, Setembro 1987.

[Henry81a] Henry, Sallie & Kafura, Denis & Harris, K. : "On the Relationship Among Three Software Metrics", Performance Evaluation Rev., Vol. 10, nº1, p. 81-88, 1981.

[Henry81b] Henry, Sallie & Kafura, Denis : "Software Structure Metrics Based on Information Flow", IEEE Transactions on Software Engineering, Vol. 7, Nº5, p 510-518, Setembro 1981.

[Henry84] Henry, Sallie & Kafura, Denis : "The Evaluation of Software Systems' Structure Using Quantitative Software Metrics", Software - Practice and Experience, Vol. 14, nº 6, p. 561-573, Junho 1984.

[Inglis86] Inglis, James : "Standard Software Quality Metrics", AT&T Technical Journal, Vol. 65, Nº2, p.113-118, Março/Abril 86.

[ISO 9126] "Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for their use", International Standards Organization, 1991.

[ISO9000-3] "Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software", International Standards Organization, Junho 1991.

[ISQAA89] "ISQAA Metrics Handbook", Information Systems Quality Assurance Association, 1989.

[Itakura82] Itakura, M. & Takayanagi, A. : "A model for estimating program size and his evaluation", Actas da *6th International Conference on Software Engineering*, p.104-109, Setembro 1982.

[Jannasch92] Jannasch, Helen : "Comparative Description of Software Quality Measurement Tools", Siemens AG, Fevereiro 1992.

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

[Kafura81] Kafura, D. & Henry, S. : "Software Quality Metrics Based on Interconnectivity", Journal of Systems and Software, Vol. 2, nº2, p. 121-131, Junho 1981.

[Kafura85] Kafura, D. & Canning, J. : "A Validation of Software Metrics Using Many Metrics and Two Resources", Actas da *8th International Conference on Software Engineering*, p. 378-385, IEEE Computer Press, 1985.

[Kafura87] Kafura, D. & Reddy, G.R. : "The Use of Software Complexity Metrics in Software Maintenance", IEEE Transactions on Software Engineering, Vol. 13, nº3, p.335-343, Março 1987.

[Kirakowski92] Kirakowski, Jurek & Porteus, Murray & Corbett, Mary : "How to Use the Software Usability Measurement Inventory: the Users´ View of Software Quality", Actas da *3rd European Conference on Software Quality*, Madrid, Novembro 1992.

[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.

[Lassez81] Lassez, J.L. & Knijff, D. Van der & Shepherd, J. & Lassez, C. : "A Critical Examination of Software Science", Journal of Systems and Software, Vol. 2, nº2, p. 105-112, Junho 1981.

[Lehder88] Lehder, Wilfred E., Smith, D.Paul & Yu, Weider D. : "Software Estimation Technology", AT&T Technical Journal, Julho/Agosto 1988.

[Levitin86] Levitin, A.V. "How To Measure Software Size And How Not To", Actas da *COMPSAC 86*, p. 314-318, IEEE Computer Society Press, Outubro 1986.

[Li87] Li, H.F. & Cheung, W.K. : "An Empirical Study of Software Metrics", IEEE Transactions on Software Engineering, Vol. 13, nº6, p. 697-708, Junho 1987.

[Linkman91] Linkman, Sue G. & Walker John G. : "Controlling Programmes Through Measurement", Information and Software Technology, Vol. 33, nº1, p. 93-102, Janeiro/Fevereiro 1991.

[Mannino92] Mannino, Phoebe & Stoddard, Bob & Sudduth, Tammy : "Análisis de la Complejidad del Software McCabe como Herramienta de Diseño y de Prueba", UNIX Magazine (versão espanhola), nº9, p.48-60, Setembro 1992.

[McCabe76] McCabe, Tom. J. : "A Complexity Measure", IEEE Transactions on Software Engineering, Vol. 2, Nº 4, p. 308-320, Dezembro 1976.

[Meredith92] Meredith, Denis C. : "Applied Software Measurement", Seminário da Advanced Technology International, Londres, Outubro 1992.

[Mills88] Mills, Everald E. : "Software Metrics", Curriculum Module SEI-CM-12-1.1, Software Engineering Institute, Carnegie Mellon University, Dezembro 1988.

[Musa75] Musa, J.D. : "A Theory of Software Reliability and Its Application", IEEE Transactions on Software Engineering, Vol. 1, Nº3, p. 312-327, Setembro 1975, reimpresso em [Basilio80], p. 194-212.

[Musa80] Musa, J.D. : "Software Reliability Measurement", Journal of Systems and Software, Vol. 1, nº3, p. 223-241, 1980, reimpresso em [Basilio80], p. 194-212.

[Musa87] Musa, J.D. & Iannino, A. & Okumoto, K. : "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, New York, 1987.

[Myers75] Myers, G. : "Reliable Software Through Composite Design", Petrocelli/Charter, New York, 1975.

[Myers77] Myers, G. : "An Extension to the Cyclomatic Measure of Program Complexity", ACM SIGPLAN Notices, Vol. 12, nº10, p. 61-64, Outubro 1977.

[NBS82] National Bureau of Standards : "Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric", Special Publication SP 500-99, 1982.

[Parkinson89] Parkinson, John : "The Case Directory", International Business Communications (IBC), Financial Publishing Ltd, ISBN 1 85271 104 3, 1989.

[Paulk91] Paulk, Mark C. & Curtis, Bill & Chrissis, Mary Beth & outros : "Capability Maturity Model for Software", CMU/SEI-91-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Agosto 1991.

[Paulk92a] Paulk, Mark C. & Humphrey, Watts S. & Pandelios, George J. : "Software Process Assessments: Issues and Lessons Learned", Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1992.

[Paulk92b] Paulk, Mark C. : "U.S. Quality Advances: The SEI's Capability Maturity Model", Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1992.

[Perez92] Perez, Ignacio & Ferrer, Pedro L. & Fernandez, Antonio : "Application of Metrics in Industry - AMI", Actas da *3rd European Conference on Software Quality*, Madrid, Novembro 1992.

[Peters82] Peters, T. & Waterman, R. : "In Search of Excellence", Harper & Row, New York, 1982.

[Potier82] Potier, D. & Albin, J.L. & Ferreol, R. & Bilodeau, A. : "Experiments with Computer Software Complexity and Reliability", Actas da *6th International Conference on Software Engineering*, p. 94-103, IEEE Computer Press, Setembro 1982.

[Pyramid91] Pyramid Consortium : "Quantitative Management: Get a Grip on Software", Y91100-4, ESPRIT Project 5425 (Pyramid), C.E.C. DGXIII, Dezembo 1991.

[Rombach87] Rombach, H.D. : "A Controlled Experiment on the Impact of Software Structure on Maintainability", IEEE Transactions on Software Engineering, Vol. 13, nº3, p.344-354, Março 1987.

[Rook90] Rook, P. (editor) : "Software Reliability Handbook", Elsevier Scientific, 1990.

[Ruston79] Ruston, H. (Workshop Chair), : "An Assessment of the State of the Art", Actas da *Workshop on Quantitative Software Models for Reliability, Complexity and Cost*, IEEE Computer Society Press, 1979.

[Shen83] Shen, V. Y. & Conte, S. D. & Dunsmore, H. E. : "Software Science Revisited: A Critical Analysis of the Theory and Its Empirical Support", IEEE Transactions on Software Engineering, Vol. 9, Nº2, p.155-165, Março 1983.

[Shen85] Shen, V.Y. & Yu, T. J. & Thebaut, S.M. & Paulsen, L.R. : "Identifying Error-Prone Software - An Empirical Study", IEEE Transactions on Software Engineering, Vol. 11, nº4, p. 317-324, Abril 1985.

[Stetter84] Stetter, F. : "A Measure of Software Complexity", Computer Languages, Vol. 9, nº3-4, p. 203-208, 1984.

[Symons88] Symons, Charles R. : "Function point analysis: difficulties and improvements", IEEE Transactions on Software Engineering, Vol.14, Nº1, 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.

[Takahashi89] Takahashi, Muneo & Kamayachi, Yuji : "An Empirical Study of a Model for Program Error Prediction", IEEE Transactions on Software Engineering, Vol. 15, Nº1, p.82-86, Janeiro 1989.

[TickIT92] TickIT Project Office : "A Guide to Software QMS Construction using ISO9001 / EN29001 / BS5750", Versão 2, Department of Trade and Industry", Fevereiro 1992.

[Troy81] Troy, D.A. & Zweben, S.H. : "Measuring the Quality of Structure Designs", Journal of Systems and Software, Vol. 2, nº2, p. 113-120, Junho 1981.

[Woodfield81] Woodfield, S.N. & Shen, V. Y. & Dunsmore, H.E. : "A Study of Several Metrics for Programming Effort", Journal of Systems and Software, Vol.2, nº2, p. 97-103, Junho 1981.

[Woodward79] Woodward, M.R. & Hennell, M.A. & Hedley,D. : "A Measure of Control Flow Complexity in Program Text", IEEE Transactions on Software Engineering, Vol.5, nº1, p.45-50, Janeiro 1979.

[Yau80] Yau, S.S. & Collofello, James S. : "Some Stability Measures for Software Maintenance", IEEE Transactions on Software Engineering, Vol. 6, nº6, p. 545-552, Novembro 1980.

[Yau85] Yau, S.S. & Collofello, James S. : "Design Stability Measures for Software Maintenance", IEEE Transactions on Software Engineering, Vol. 11, nº9, p. 849-856, Setembro 1985.

[Yin78] Yin, B. & Winchester, J. : "The Establishment and Use of Measures to Evaluate the Quality of Software Designs", Actas da *Software Quality and Assurance Workshop*, New York, Association for Computing Machinery, 1978.

[Yu90] Yu, Weider D. & Smith, D. Paul & Huang, Steel T. : "Software Productivity Measurements", AT&T Technical Journal, Maio/Junho 1990.