PUC

Série: Monografias em Ciência da Computação
Nº 24/77

FATORES HUMANOS NA PROGRAMAÇÃO

por

José A. Chahon
José A. F. Mendes
Nelson Soley

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vivente, 225 – ZC-19
Rio de Janeiro – Brasil

Série: Monografias em Ciência da Computação
Nº 24/77

Editor da Série: Michael F. Challis

Dezembro, 1977

FATORES HUMANOS NA PROGRAMAÇÃO

por

José A. Chahon **
José A. F. Mendes **
Nelson Soley **

* Trabalho patrocinado em parte pels FINEP.

** Alunos de pós-graduação do Deptº de Informática - PUC/RJ.

Para obter cópias dirija-se a:

Rosane T.L. Castilho
Chefe, Setor de Doc. & Inf.
Dept9 Informática - PUC/RJ
Rua Marquês de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - BRASIL.

Abstract

Human factors involved in the programming activity are considered, with a special emphasis on aspects often disregarded both in the literature and in the practical world.

Keywords

Human factors, aptitude tests, programming teams, large software projects.

Resumo

São considerados fatores humanos envolvidos na atividade de programação, com ênfase especial em aspectos frequentemente negligenciados na literatura e na prática.

Palavras Chave

Fatores humanos, testes de aptidão, equipes de programação, projetos grandes de software.

SUMÁRIO

FATORES HUMANOS NA PROGRAMAÇÃO

1. INTRODUÇÃO

A programação de computadores é uma atividade humana. Não há dúvidas quanto a isto. Entretanto, muitos programadores nunca encararam a programação sob este ponto de vista, talvez por se acharem demasiadamente presos a aspectos puramente técnicos do problema.

Este trabalho procura chamar a atenção para os fatores humanos envolvidos na programação e que têm sido injustamente desprezados tanto na literatura quanto na prática.

2. LIMITAÇÕES IMPOSTAS AO PROGRAMADOR

Caso se deseje estudar programação como uma atividade humana, antes de mais nada é necessário que se crie medidas para determinar o desempenho dos programadores e dos programas, de maneira que seja viável fazer-se comparações entre dois programadores ou entre dois programas.

Os aspectos envolvidos mais importantes são:

a. Especificações

De todas as exigências impostas a um programa, esta é a mais importante. As especificações de um programa têm de ser atendidas.

Em outras palavras, "um programa que funciona é sempre melhor que outro que não funciona".

b. Cronograma

Esta é outra exigência importante, embora muitas vezes não seja atendida. Um atraso no cronograma pode significar, em alguns casos, perdas elevadas, além de criar ressentimentos entre usuário e desenvolvedor.

c. Adaptabilidade

A maioria dos programas desenvolvidos por programadores profissionais sofre modificações durante seu ciclo de vida. Sendo assim, estes programas devem ser planejados e desenvolvidos tendo em conta este fator. A documentação do programa também é importante quanto a este aspecto.

d. Eficiência

A eficiência de um programa é um aspecto importante, que deve sempre ser levado em conta. Entretanto, eficiência geralmente implica perda de adaptabilidade e portabilidade. Além disso, é difícil se medir a real eficiência de um programa.

Em resumo, não é uma questão simples determinar-se a qualidade de um programa.

3. COMO MEDIR A QUALIDADE DE PROGRAMAS E PROGRAMADORES

A programação é uma atividade rica e complexa e para se estudá-la é preciso que os métodos usados também o sejam.

Para que seja possível fazer um estudo válido é necessário se evitar problemas que possam surgir, tais como:

a. Observações usando-se pequena quantidade de dados

b. Observação das variáveis erradas

c. Interferência com o fenômeno observado

d. Muitos dados e poucas informações

e. Experiências com muitas restrições

f. Não estudar efeitos de grupo

g. Medições apenas do que é fácil medir

h. Usar precisão não justificável

i. Usar apenas treinandos.

4. PROGRAMAÇÃO EM GRUPO

Os programadores geralmente não trabalham sozinhos. Mesmo quando um programador é o único responsável pelo desenvolvimento de um programa, muitas vezes ele tirará dúvidas com outros ou vice-versa.

É prejudicial para o programador trabalhar sozinho, como nós veremos. Quando ele trabalha com outros, muitos tipos de relações podem surgir.

Pode se identificar três tipos de agrupamentos de programadores - o grupo, a equipe e o projeto.

O grupo de programação é um conjunto de programadores, trabalhando no mesmo lugar, mas trabalhando em programas separados (embora possa haver relação entre alguns dos programas).

A equipe de programação é um conjunto de programadores que estão desenvolvendo um único programa.

Um projeto é um grupo de programadores mais as atividades de suporte. O objetivo é desenvolver um sistema integrado ou programas bastante interrelacionados. Há ainda um gerente do projeto e uma organização burocrática.

4.1 O Grupo de Programação

4.1.1 Introdução

O estudo do grupo de programação é importante para se entender os outros tipos de organização para programação. Mesmo quando existe uma organização formal dos programadores em equipes e projetos, surgem ligações informais de maneira semelhante àquelas do grupo "não estruturado". Estas ligações informais muitas vezes têm um efeito positivo bastante grande na produtividade do grupo, devido à troca de informações e experiências que proporciona.

Outro ponto importante é a relação que existe entre a organização social e o ambiente físico. Como o "layout" das salas onde os programadores trabalham afeta a interação social

entre eles, que por sua vez afeta o trabalho que está sendo feito. Um exemplo simples: divisórias, que se acham colocadas em muitas salas. Elas impedem toda a comunicação útil, mas não impedem o barulho de penetrar.

4.1.2 Programação: uma atividade individual?

Muitos programadores e gerentes defendem a idéia que a programação é uma atividade individual e que só depende da habilidade do programador. A maioria dos programadores possivelmente preferem trabalhar sozinhos que em grupo.

Esta idéia é muito provavelmente a maior barreira que existe para a melhoria da qualidade de programação.

Grande parte dos programadores são do tipo que "desejam trabalhar sozinhos para poderem ser criativos". Embora eles estejam afastados das pessoas, eles estão forçosamente presos a seus programas. Alguns programadores acabam por considerar os programas que eles desenvolvem como propriedade individual, como os "seus" programas. Esta atitude faz com que estes programadores não vejam erros clamorosos nos "seus" programas, porque eles analisam os resultados superficialmente (já que são os únicos responsáveis pelos erros do programa e são poucos os que admitem que cometeram um erro).

Assim sendo, caso se pretenda melhorar a qualidade da programação, para que os programas realmente satisfaçam as especificações, é necessário que se criem estruturas organizacionais que evitem a programação individual.

4.1.3 Programação aberta

O problema do ego em programação só pode ser superado através de uma reestruturação do ambiente social, que por sua vez provocará uma reestruturação no sistema de valores dos programadores. Na "programação aberta", todos os programas desenvolvidos por cada programador são revistos e criticados por pelo menos uma outra pessoa.

Os programadores devem ser treinados para aceitar sua condição de seres humanos - ou seja, aceitar a impossibilidade de

funcionarem como máquinas. Dijkstra, no livro STRUCTURED PROGRAMMING (ref. o), pg. 1 (On our inability to do much) trata deste problema, que é a causa do fracasso de muitos projetos.

Textualmente, ele diz: "As a slow-witted human being I have a very small head and I had better learn to live with it and to respect my limitations and give them full credit, rather than to try to ignore them, for the latter vain effort will be punished by failure".

Dijkstra repete este mesmo argumento em sua 1972 ACM Turing Award Lecture (ref. p): "The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague".

Porque a "programação aberta" não é muito difundida? Uma razão importante é que os próprios programadores difundem a idéia que programação é uma atividade individual para alguns superdotados (por exemplo, ver referência f). Outra razão é que nunca foi feita uma comparação entre os dois tipos de programação em termos de qualidade dos programas produzidos. Algumas experiências sobre fatores que influenciam a produtividade dos programadores já foram feitas, mas infelizmente deu-se ênfase aos aspectos puramente mecânicos de programação, deixando-se de lado os aspectos sociais.

Uma vantagem da "programação aberta" é que o tempo de depuração dos programas é reduzido. Além disso, como existe mais de uma pessoa familiarizada com o programa sendo desenvolvido, torna-se mais fácil estimar o progresso realmente feito. A adaptabilidade dos programas também aumenta, já que pelo menos duas pessoas sabem o que um dado programa faz. Finalmente, como os programadores têm de ler programas escritos por outros, a qualidade deles tende a aumentar.

5. A EQUIPE DE PROGRAMAÇÃO

A falta de experiência em programação torna-se mais evidente quando o tamanho do sistema a ser produzido aumenta. Há uma diferença social entre programas grandes e pequenos. En-

quanto todos os detalhes de um programa podem ser guardados na cabeça de uma única pessoa, não há necessidade de coordenação do esforço de programação. A interação entre dois programadores que estão revendo um programa que ambos poderiam ter escrito é completamente diferente da interação entre dois programadores trabalhando em partes diferentes de um todo, o qual é grande demais para qualquer deles desenvolver isoladamente. A diferença está na maneira pela qual exigências conflitantes são resolvidas. No primeiro caso, a resolução do conflito é feita por uma pessoa, talvez ajudada pela outra, mas com todo o processo sob seu controle exclusivo. No segundo caso, as exigências técnicas conflitantes podem dar origem a conflitos entre as pessoas envolvidas. Um mecanismo social tem de ser formado para resolver estes conflitos de melhor maneira.

5.1 Como uma equipe é formada

Uma equipe de programação é formada para fazer um trabalho cujas exigências são demasiadas para uma única pessoa, não só em termos de tempo como em termos de qualificação técnica.

Dado um sistema para ser desenvolvido, há uma capacidade técnica mínima e um tempo mínimo necessário para desenvolvê-lo. Em outras palavras, quase todos os sistemas podem ser desenvolvidos com programadores de menor qualidade, desde que estejamos dispostos a aceitar um tempo de desenvolvimento maior (e não estejamos abaixo da qualificação técnica mínima indispensável).

A regra básica para se formar uma equipe parece ser esta, para se obter o melhor sistema pelo menor custo - dê aos seus melhores programadores, para desenvolver um dado sistema, o prazo que minimiza o número de programadores necessários para desenvolver o sistema. Quando o prazo é menor ou os programadores menos experimentados, custos e incertezas aumentarão fatalmente. A pior maneira para se desenvolver um sistema, é contratar uma horda de programadores inexperientes e colocá-los para trabalhar sob pressão e sem supervisioná-los.

Quando um grupo tem programadores pouco experientes, o objetivo da equipe passa a ser duplo (produção e treinamento) e a produção é afetada negativamente.

Em programação, o modo de organização de uma equipe é fortemente influenciado por dois fatores - a organização do sistema a ser desenvolvido e a composição da equipe. Como frequentemente existem várias maneiras de se estruturar o desenvolvimento de um sistema, mas raramente há possibilidade de se ter mais de um grupo para tal desenvolvimento, a estruturação é normalmente feita de modo a se adaptar às qualidades e defeitos dos membros da equipe. Isto está completamente errado. O ideal seria escolher-se a estrutura do sistema e só então montar um grupo que atendesse àquela estrutura.

5.2 Definição de Objetivos

Uma das vantagens da "programação aberta" é que cada membro da equipe tem oportunidade de examinar o trabalho de todos os demais. Isto tende a evitar o estabelecimento de uma hierarquia muito pronunciada.

Sentimentos recalcados por parte de um membro da equipe sobre a inferioridade de sua tarefa em relação às demais pode causar grandes prejuízos ao esforço comum. A "programação aberta" tende a moderar tais sentimentos, porque cada programador tem maior participação no desenvolvimento do sistema em comparação com a programação individual. Ainda assim, caso uma equipe comece a trabalhar antes que exista um consenso a respeito da estrutura do projeto e da divisão do trabalho, surgirão problemas inevitavelmente.

Psicólogos sociais já comprovaram em outras situações que se um ou mais membros não concordam com os objetivos da equipe, o desempenho desta equipe diminuirá (não só porque estes membros rendem menos, mas também porque outros membros percebem o problema e têm seu rendimento afetado).

Para conseguir um consenso real sobre os objetivos de uma equipe, a melhor maneira é deixar que a própria equipe os estabeleça. Participação no estabelecimento dos objetivos faz com que eles sejam mais claramente compreendidos. Além disso, a simples participação no estabelecimento dos objetivos é um fator importante na aceitação, por cada membro da equipe, destes mesmos objetivos.

Evidentemente, há restrições quanto ao estabelecimento de objetivos. Uma equipe é usualmente constituída para realizar uma tarefa que foi provavelmente estabelecida mesmo antes da equipe ser reunida. Mas as tarefas de programação não são comumente definidas de maneira muito precisa de imediato, de modo que os membros da equipe têm oportunidade de definir muitos objetivos, mesmo que secundários. Os gerentes devem evitar interferência excessiva neste aspecto, para não diminuir o entusiasmo do grupo.

5.3 Liderança na Equipe

Liderança, segundo os cientistas sociais, é a capacidade de influenciar pessoas. Programadores, sendo pessoas que dão valor ao trabalho criativo e à competência profissional, tendem a confiar em pessoas que sejam competentes em programação.

Os gerentes de equipes de programação, caso não saibam programar, terão dificuldades pela frente, a menos que reconheçam sua incapacidade técnica.

Numa equipe organizada democraticamente, a liderança não se restringe a uma única pessoa, mas muda de membro para membro de acordo com as necessidades do desenvolvimento. A liderança, entretanto, não é distribuída igualmente numa equipe democrática. O fator importante é que estas lideranças são baseadas nas realidades internas da equipe e não impostas de fora.

Infelizmente, nenhuma equipe de programação pode ser completamente democrática no sentido de poder sempre escolher a pessoa certa para liderá-los em cada situação. Primeiramente, a pessoa certa pode não pertencer à equipe. Além disso, a escolha feita pode ser errônea.

6. O PROJETO DE PROGRAMAÇÃO

Quando duas equipes precisam trabalhar juntas para atingir algum objetivo, surge uma função de coordenação. Novos tipos de relações sociais aparecem. Por exemplo, dois programadores interagem tanto como indivíduos quanto como membros de equipes diferentes - agora os objetivos de suas equipes podem dife

rir, mesmo que seus objetivos individuais sejam muito semelhantes. O sucesso do membro de uma equipe é medido não só com relação a outros membros da equipe, mas também como parte componente de uma equipe que por sua vez é comparada com outras.

6.1 Estrutura de um Projeto

Em um projeto de programação há necessidade de se separar o trabalho propriamente dito da avaliação da quantidade de trabalho já produzida. Possivelmente a maior fraqueza da organização hierárquica está exatamente neste ponto, porque ela não permite esta separação. Este tipo de organização geralmente faz com que haja uma deterioração na qualidade das informações que fluem por ela. Para combater esta deterioração, surgiram modificações na estrutura hierárquica simples.

Muitos projetos têm um grupo de padrão cuja principal atividade não é produção, mas avaliação da produção de outros grupos. Quando atinge-se o estágio de testes, o grupo de padrões (ou parte dele) pode tornar-se responsável pelo teste do sistema. Grupos especiais podem também serem criados para outras funções, tais como um grupo de hardware, um grupo de especialistas em documentação, etc. Um tipo de grupo, que é particularmente útil em certas situações, é um grupo de relações públicas para proteger os outros grupos de interferências externas. Há ainda a possibilidade de grupos especiais, de duração limitada, terem de ser criados. Por exemplo, um erro de programação que pelos métodos usuais não é descoberto, pode provocar a criação de um grupo especial para localizá-lo. Para poder estabelecer tais grupos temporários, o gerente deve ter uma certa folga de pessoal. Nenhum projeto, por mais cuidadosamente planejado que seja, pode antecipar todas as situações que possam surgir. Assim, um projeto típico geralmente sofre várias reorganizações durante o seu ciclo de vida.

Tais reorganizações podem provocar problemas. Algumas pessoas podem se recusar a mudar de grupo, a trabalhar com uma certa pessoa ou em um certo trabalho. Ou pode surgir o problema inverso, ou seja, alguns insistirem em trabalhar com uma certa pessoa, ou em um certo grupo ou em um certo trabalho. Estes problemas surgem, normalmente, de uma divisão de lealdades.

Cada pessoa trabalhando em uma equipe dentro do projeto passa a ver alguns dos objetivos da equipe como seus. Porém, quando a equipe realizar uma tarefa específica, os objetivos da equipe não coincidem necessariamente com os objetivos globais do projeto ou com os objetivos de outro grupo. Estes conflitos invariavelmente se manifestam nas relações sociais. Por exemplo, certos grupos são considerados menos importantes que outros. Um grupo de documentação é geralmente considerado pouco importante pelos grupos de programação do projeto.

Este tipo de divisão é muitas vezes encorajado pelos gerentes, por eles também terem seus preconceitos e permitirem às vezes que isto apareça na estrutura do projeto.

Qualquer grupo de teste geralmente tem problemas de relacionamento com os outros grupos, porque a tarefa deles é criticar e poucos são os que recebem críticas sem se aborrecer.

Cabe aos gerentes minimizarem todos estes problemas, tratando todos os grupos da mesma maneira e sem parcialidades.

6.2 Problemas Sociais em Projetos Grandes

A grande distância entre a cúpula do projeto e os programadores é a causa de muitos problemas sociais em projetos grandes. Há apenas duas funções em uma equipe de programação: a de programador ou chefe dos programadores.

Dentro de um projeto, entretanto, aparece uma terceira função: o chefe de chefes. A diferença entre os dois níveis de chefia é grande, porque no primeiro nível o chefe ainda tem algum contacto com o trabalho sendo desenvolvido, enquanto o chefe do segundo nível acompanha o trabalho indiretamente, através das informações dos outros chefes.

Os gerentes de nível mais alto embora originalmente possam ter sido bons programadores, quando atingem estes pontos já perderam praticamente toda a capacidade anterior. Assim sendo, eles tendem a recompensar os programadores não pelo trabalho que eles fazem, mas pela aparência de trabalho. Os programadores que chegam cedo, por exemplo, são considerados melhores que aqueles que costumam chegar atrasados. Os programadores que trabalham

depois do expediente possivelmente não serão recompensados, pois os supervisores já se retiraram para casa.

Provavelmente a atitude mais correta de um supervisor é fazer com que os chefes de primeiro nível realmente controlem o trabalho desenvolvido em seu grupo, inclusive lendo os programas caso necessário.

Os supervisores, como não podem estabelecer sua superioridade técnica sobre os programadores, tendem a confiar em símbolos de "status" para mostrar esta autoridade. Por exemplo, alta prioridade no computador, um terminal próprio, secretárias, etc. Isto ocasiona gastos supérfluos muitas vezes, servindo apenas para encarecer o projeto.

Um outro problema social em projetos grandes é a discriminação contra as mulheres. Mesmo que uma mulher seja a programadora mais competente e de maior produtividade, dificilmente ela será apontada para chefia de seu grupo. Evidentemente, esta situação prejudica o desenvolvimento do projeto. Em um projeto de programação, a exclusão de qualquer pessoa de qualquer posição por qualquer razão que não seja falta de competência, prejudica o desempenho do projeto.

6.3 Determinação de Desempenho

Uma consequência do tamanho de um projeto é que fica mais difícil medir o desempenho. Parte da dificuldade vem do fato que é impossível para uma única pessoa julgar todas as outras. Como o acompanhamento do desenvolvimento de um programa é uma atividade muito subjetiva, as opiniões divergem quando se trata de determinar em que ponto um sistema em desenvolvimento se acha. Quando se reune todas as opiniões emitidas sobre diferentes programas por diferentes pessoas para se obter o estado atual do desenvolvimento, as possibilidades de erros são muitas.

Há uma tendência de se eliminar nos relatórios de progresso as variações grandes, de modo a transmitir ao nível de chefia imediatamente superior a idéia de que tudo está correndo bem.

Assim, o mais alto nível de chefia recebe relatórios que pouco espelham o que está ocorrendo a nível dos programadores. Existirá algum modo de se evitar este problema de "filtragem"? Uma possibilidade é de se indicar um "advogado do diabo" para tentar diminuir a tendência de se ter opiniões que estejam de acordo com o esperado. Como o papel do "advogado do diabo" é oficializado, a pressão que ele sofre é menor do que seria em qualquer outro membro do grupo que tivesse uma opinião diferente dos demais. Em alguns projetos, este papel é desempenhado por diferentes pessoas ao longo do tempo para tornar mais efetiva a atuação do "advogado do diabo". Outra possibilidade é se ter uma estrutura menos hierarquizada, em que exista menos pressões. Mesmo assim, não se consegue eliminar totalmente as pressões sobre aqueles que têm de fazer os relatórios de progresso. Estas pressões partem agora principalmente dos elementos que estão no mesmo nível que o relator e cujo desempenho está sendo descrito no relatório.

7. A PROGRAMAÇÃO COMO UMA ATIVIDADE INDIVIDUAL

Vamos procurar agora verificar quais são os fatores individuais que afetam a programação, fatores estes que os psicólogos chamam de "variações individuais", ou seja, quando duas pessoas recebem exatamente a mesma tarefa para executá-la no mesmo ambiente, as diferenças de comportamento podem ser atribuídas a estes fatores. As variações individuais que mais nos interessam podem ser classificadas em quatro categorias: personalidade, inteligência, treinamento e experiência.

7.1 Programador Profissional Versus Programador Amador

Existem várias diferenças importantes entre os dois. Talvez as maiores diferenças estejam relacionadas com os usuários dos programas. Quase sempre o usuário de um programa feito por um programador amador é ele próprio, enquanto o profissional escreve programas para outros usarem.

Há muitos anos, quando os sistemas operacionais eram rudimentares, a diferença entre o profissional e o amador não era tão

pronunciada. Hoje, entretanto, muitas das coisas que os amadores desejam são feitas automaticamente pelos sistemas operacionais. Dessa maneira, a distância entre os programadores profissionais e amadores aumentou. Paradoxalmente, entretanto, os amadores têm cada vez menos consciência deste fato, já que eles se tornam cada vez menos conscientes do que os sistemas operacionais fazem para eles.

Outra diferença é que o amador está aprendendo assuntos relacionados com seu problema e qualquer aprendizado a respeito de programação é um mero acidente. O profissional, por outro lado, está aprendendo fatos relacionados com sua profissão - programação - enquanto que o problema sendo programado tem pouco interesse para ele.

7.2 Influência dos Fatores de Personalidade

Embora não exista uma definição de personalidade universalmente aceita, vamos adotar a seguinte:

- "Personalidade é a integração de todas as características de um indivíduo numa organização única que determina, e é modificada por suas tentativas de adaptação a um ambiente que muda continuamente".

Podemos verificar desta definição que a personalidade de uma pessoa não é fixa. Entretanto, a personalidade não muda quando não há razão para ela mudar.

7.2.1 Invariantes na Personalidade

Se a personalidade "é modificada por ... tentativas de adaptação", pode-se usar as mudanças de personalidade como sinal que o ambiente mudou. Mas mudanças na personalidade - embora elas podem representar situações extremamente sérias - não são frequentes. O que mais interessa é como as características mais ou menos invariantes da personalidade de um programador afetam seu rendimento.

Um modo de se caracterizar a personalidade de uma pessoa é através de seus traços, tais como imaginação, estabilidade emocional, astúcia, etc.

Tentativas de se colocar uma pessoa no trabalho que melhor se adapta a sua personalidade são sujeitas a falhas. Isto porque a personalidade contém várias "camadas" - e uma mesma camada exterior pode ter interiores bem diferentes. Evidentemente, deve-se ter cautela ao inferir as camadas internas da personalidade a partir de observações da camada externa.

Outro fator que distorce nossa percepção da personalidade de alguém é que muitas pessoas tendem a se comportar diferentemente com seus chefes, com seus colegas e seus subordinados. Muitas pessoas comportam-se diferentemente por outras razões, tais como idade e sexo.

7.2.2 Traços de Personalidade Críticos

Medir a personalidade é difícil e adaptar personalidades a descrições de tarefas é talvez impossível. Não haverá, entretanto, um modo de se selecionar aquelas pessoas cujas personalidades as tomam apropriadas para programação?

A maneira pela qual a personalidade interage com o sucesso em programação é sutil, mas pode-se talvez fazer algumas afirmativas a respeito dos traços de personalidade que podem conduzir ao fracasso na programação. Uma pessoa que não tenha capacidade para tolerar situações de tensão, por um período de uma semana ou mais, dificilmente poderá ser um bom programador profissional.

Devido a diversidade do trabalho em programação, as pessoas que não têm uma certa capacidade para se adaptar a mudanças rápidas provavelmente terão dificuldades como programadores profissionais.

Além disso, o programador profissional deve ser organizado. Há centros de computação, por exemplo, que selecionam seus programadores através de um teste em que o candidato escolhido é aquele que apresenta a prova mais bem organizada, independentemente do resultado que ele obtém.

Outros traços de personalidade convenientes para o programador são humildade, força de caráter e senso de humor.

8. TESTES DE APTIDÃO PARA PROGRAMADORES

Muitas organizações usam testes de aptidão para selecionar seus programadores. Os resultados destes testes, segundo alguns, são válidos e são indicadores confiáveis do potencial dos programadores.

Um dos testes mais populares é um teste desenvolvido pela IBM e que consta de três partes: série de letras (ou números), séries de figuras e raciocínio aritmético.

Outros testes verificam também a capacidade para analisar diagramas e extensão do vocabulário do programador ou candidato a programador.

O objetivo destes testes é tentar predizer o sucesso no treinamento e/ou trabalho. Como outros fatores, tais como motivação e perseverança são importantes para o sucesso no trabalho, é pouco provável que estes testes sejam suficientes.

Um problema com estes testes tradicionais é que existem, embutidos neles, influências da cultura e da língua. Assim sendo, algumas companhias e pesquisadores desenvolveram testes não-verbais e não-matemáticos, que não têm limitações culturais, ocupacionais ou verbais. Foi sugerido, por exemplo, o uso de um teste, sem limitação de tempo, usando apenas figuras, com vinte questões (cf. ref. d).

Este teste apresenta alguns problemas, no entanto. Um deles, por exemplo, é que não é possível provar que cada questão tem apenas uma resposta correta.

Há necessidade de mais pesquisas nesta área, pois os testes atuais não parecem satisfatórios.

9. MOTIVAÇÃO, TREINAMENTO E EXPERIÊNCIA

Os psicólogos já mostraram que o desempenho humano em uma dada tarefa é função da própria tarefa e como a pessoa entende a tarefa. O desempenho também será afetado por diferenças individuais tais como personalidade e inteligência. Embora a personalidade pode ser mudada e a inteligência pode ser aumentada um pouco, uma melhoria real de desempenho só pode vir do treinamen-

to e experiência. Além disso, o desempenho de uma pessoa é alterado pela motivação, que é definida como uma "força motora interna".

Um dos resultados mais conhecidos e aceitos das pesquisas sobre motivação é que, aumentando a "força motora" o desempenho inicialmente aumenta, atingindo um máximo e a partir daí, aumentando-se a "força motora" o desempenho cai rapidamente para zero. Esta queda rápida no desempenho é especialmente observável em tarefas complexas e é, portanto, de especial interesse para a programação. Tentar achar de qualquer maneira um erro num programa é contraproducente. Pressionar um programador para a rápida eliminação de um erro pode ser a pior estratégia possível - é, entretanto, a estratégia mais comum.

Do ponto de vista do gerente de programação, ou do próprio programador, a primeira pergunta importante a respeito da motivação é: em que grau o programador já está motivado? A resposta a esta pergunta permite saber se deve-se aumentar ou diminuir a "força motora". Normalmente os programadores são supermotivados, o que permite explicar porque tantos projetos de programação se desintegram quando a pressão aumenta.

Normalmente, os fatores de motivação de um programador são: um aumento salarial, envolvimento pessoal no planejamento das tarefas e promoções.

Além da motivação, outra grande influência no desempenho de um programador é o treinamento ou, se este é suficientemente geral, a educação. Pouco se sabe a respeito da educação de programadores, embora quantias enormes tenham sido gastas em treinamentos. Talvez se espere demasiado da educação formal. Na situação atual, os programadores aprendem mais com o computador do que em qualquer curso.

10. LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação são o nosso meio de comunicação com os computadores. Há várias diferenças entre uma linguagem de programação e uma linguagem natural.

Uma linguagem de programação não tem uma forma falada, por exemplo. Isto faz com que as linguagens de programação evoluam mais lentamente e, além disso, é difícil falar a respeito de uma linguagem de programação sem lápis e papel.

Possivelmente a maior diferença é, entretanto, o fato das linguagens naturais permitirem intercâmbio entre as pessoas que estão se comunicando, ou seja, o que uma fala a outra, se desejasse, poderia falar também. Em outras palavras: "todos nós falamos a mesma linguagem".

Quando se "fala" com o computador, no entanto, está-se falando linguagens diferentes. Felizmente, há uma tendência hoje em dia de se tentar aumentar o intercâmbio entre as pessoas e o computador, o que certamente aumentará a eficiência de programação.

Quando se tenta avaliar uma linguagem de programação de um ponto de vista psicológico, não se pode colocar toda a culpa de má programação nos programadores. Caso os mesmos programadores sempre obtenham melhores resultados com a linguagem A do que com a linguagem B, qual será a validade de se argumentar que os programadores deveriam aprender melhor a linguagem B? Evidentemente, estes fatos podem mudar ao longo do tempo, porque, qualquer que seja a linguagem, sempre se precisa aprender como usá-la. Assim sendo, para se responder a esta pergunta seria necessário levar em consideração os cursos, o material de ensino, a linguagem, mensagens do compilador, etc. Talvez, por exemplo, as mensagens que apareçam nos terminais ou listagens sejam mais motivantes em uma linguagem que na outra.

De uma certa maneira, a razão pela qual é tão difícil descobrir a fonte da ineficiência em programação é que, caso os programadores fossem ideais, não haveria necessidade de linguagens de programação. É uma dificuldade psicológica que impede que os programas sejam escritos diretamente em linguagem de máquina. Programação envolve comunicação entre duas espécies diferentes (homem e computador) e as linguagens de programação simplesmente tentam tornar a comunicação mais fácil para o homem, já que o computador não tem este problema.

11. O RELACIONAMENTO USUÁRIO-ANALISTA

É de se admitir que o estabelecimento de bons canais de comunicação entre os analistas e os usuários resultará em melhor compreensão mútua. O problema básico é determinar e implementar os passos necessários para esta melhoria. Felizmente, a maioria dos gerentes dos usuários desejam cooperar e aprender mais sobre processamento de dados. Isto é demonstrado pela freqüência com que artigos relacionados com processamento de dados aparecem em publicações para gerentes e pelo número de gerentes que participam em cursos, seminários e conferências relacionados com processamento de dados. Por outro lado, a porcentagem de gerentes de processamento de dados e analistas de sistemas que participam em cursos, seminários e conferências relacionados com gerência em geral e com sua companhia em particular não parece significante.

O problema de comunicações tem portanto dois aspectos: os gerentes dos usuários estão interessados em processamento de dados, mas os gerentes de processamento de dados não parecem estar interessados em aprender mais a respeito de gerência e de sua própria companhia.

Outro problema é o das perspectivas diferentes: o gerente de processamento de dados precisa reconhecer que o ponto de vista do usuário pode diferir substancialmente do seu. O gerente de processamento de dados e seus subordinados tendem a se envolver mais com os aspectos técnicos, tais como linguagem de programação, sistema operacional e quantas partições usar. Para falar a verdade, o usuário não se interessa em saber se seus dados são processados por um computador ou por um ábaco: ele simplesmente deseja resultados.

Os usuários consideram o processamento de dados como um serviço que dá informações. Um usuário aceita a necessidade e responsabilidade pela preparação dos dados de entrada para o computador e em troca espera receber informação acurada a tempo.

Olhando-se as atividades e interesses do gerente de processamento de dados e os interesses dos gerentes dos usuários,

parece que o principal interesse dos usuários - produção de informação acurada a tempo - é a atividade na qual o gerente de processamento de dados gasta menos tempo.

A menos que instalação de processamento de dados seja muito boa, a adição de novas aplicações ou a saída de uma ou duas pessoas chaves podem resultar em mal funcionamento da instalação. Uma instalação bem organizada e controlada não tem este tipo de problema.

O que os usuários desejam de um sistema é basicamente:

- desempenho bom
- projeto feito por um analista com conhecimento dos problemas que o sistema vai resolver
- saídas confiáveis e acuradas
- flexibilidade suficiente para permitir mudanças na entrada e na saída sem necessidade de reprogramação total.

Por outro lado, os usuários não devem obstruir (mesmo que involuntariamente) o desenvolvimento do sistema por falta de cooperação, requisitos não razoáveis e comunicação incompleta de suas necessidades. Estes problemas podem ser atenuados se os usuários souberem o que é um computador, como ele é usado e como ele afeta as suas tarefas.

REFERÊNCIAS

a. The Psychology of Computer Programming, GERALD M. WEINBERG
Van Nostrand Reinhold Company, 1971.

b. The Program Development Process: The Individual Programmer
JOEL D. ARON - Addison-Wesley Publishing Company, 1974

c. The People Side of Top-Down, DANIEL R. RICHARDSON
Infosystems, julho de 1975, pg. 34-35

d. Placing the Programmer, CHARLES J. TESTA e SHELDON LAUBE
Journal of Systems Management, março de 1974, pg. 31 a 35

e. The Psychology of Improved Programming Performance
GERALD M. WEINBERG
Datamation, novembro de 1972, pg. 82 a 85

f. How to be a Superprogrammer, EDWARD YOURDON
Infosystems, fevereiro de 1976, pg. 32-33

g. First, understand yourself, H.A. SHEARRING
Journal of Systems Management, julho 1974, pg. 14 a 17

h. Origins of Systems Project, T.C. WILLOUGHBY
Journal of Systems Management, outubro 1975, pg. 18 a 26

i. Motivation and Leadership
Auerbach Publishers, Inc. 1975 - 9 páginas
(DP Administration - Personnel) 2-01-02

j. Improving Relations with User Departments
Auerbach Publishers, Inc. 1976 - 15 páginas
(DP Administration - Personnel) 2-01-04

k. Managing a Programming Project, P.H.METZGER
Prentice-Hall, Inc. 1973

l. Software Engineering: Concepts and Techniques
J.M.BUXTON, PETER NAUR e B.RANDELL (editors)
Petrocelli/Charter, 1976

m. The Mythical Man-Month, F.P.BROOKS, JR.
Addison-Wesley Publishing Company, 1975

n. Practical Strategies for Developing Large Software Systems
E.HOROWITZ (editor)
Addison-Wesley Publishing Company, 1975

o. Structured Programming, O.J.DAHL, E.W.DIJKSTRA, C.A.P.HOARE
Academic Press, 1972

p. The Humble Programmer (1972 ACM Turing Award Lecture)
EDSGER W. DIJKSTRA
Communications of the ACM, Outubro 1972, pg.859 a 866

q. Managing the Creative Engineer, MICHAEL F. WOLFF
IEEE Spectrum, agosto 1977, pg. 52 a 57

r. The Primacy of the User, WILLIAM AINSWORTH
Infosystems, maio de 1977, pg. 50 a 54

s. Psychology and Program Design, DAVID FROST
Datamation, maio 1975, pg. 137-138