Medindo a Produtividade do Desenvolvimento de Aplicativos

Por Allan J. Albrecht
Proc. Joint SHARE/GUIDE/IBM Application Development Symposium (October, 1979), 83-92
IBM Corporation, White Plains, New York

Tradução do original disponível em: http://www.fattocs.com.br/artigos/MeasuringAplicationDevelopmentProductivity.pdf

Nesse texto, eu gostaria de compartilhar com vocês algumas experiências em medições de produtividade em projetos de desenvolvimento de aplicativos na unidade *DP Services* da IBM. Eu tenho vários objetivos com esse texto:

- Descrever nossa medição de produtividade; que tem sido eficaz na medição de produtividade em todas as fases do planejamento do software, incluindo a fase de projeto e tem nos habilitado a comparar os resultados de projetos que utilizam diferentes tecnologias e linguagens de programação.
- Mostrar como nós utilizamos essa medida para determinar a tendência de produtividade em nossa empresa.
- Identificar alguns fatores que afetam a produtividade e mostrar como nós determinamos a sua importância relativa.

Agora irei descrever a nossa organização. Irei falar um pouco sobre os nossos objetivos de gestão.

A unidade *DP Services* consiste em cerca de 450 pessoas contratadas envolvidas no desenvolvimento de aplicativos para clientes da IBM. Tanto os clientes quantos os empregados estão localizados nos Estados Unidos. A qualquer momento, existem cerca de 150-170 contratos em curso. Os projetos abrangem todas as indústrias.
Eles abordam o espectro dos requisitos funcionais do processamento de dados: entrada e controle de pedidos, processamento seguro de pedidos, sistemas de informações hospitalares do paciente, sistemas de controle de comunicação de dados, etc. O tamanho médio do contrato é de 2 ou 3 pessoas. O número de projetos a cada ano requer de 15 a 20 pessoas, e vários exigem de 35 a 40 empregados e clientes, trabalhando na gerência do projeto, para desenvolver a aplicação.

Cada projeto é executado de acordo com o Processo de Desenvolvimento de Aplicações da *DP Services* (figura 1). Não quero deixar a impressão de que nós realizamos todo o projeto em cada uma dessas centenas de contratos. Na maioria dos contratos, nós fazemos apenas tarefas específicas cobrindo parte das funções previstas no projeto. Nossa abordagem é uma abordagem gradual para o desenvolvimento de aplicativos. Essa abordagem gradual consiste em uma fase de projeto seguida por uma fase de implementação do sistema. O projeto do sistema é completo e aprovado antes da sua implementação ser inciada. Esse gráfico representa a distribuição do esforço de desenvolvimento nessas fases. Ele se baseia na experiência da *DP Services* nos projetos dos últimos 3 ou 4 anos.

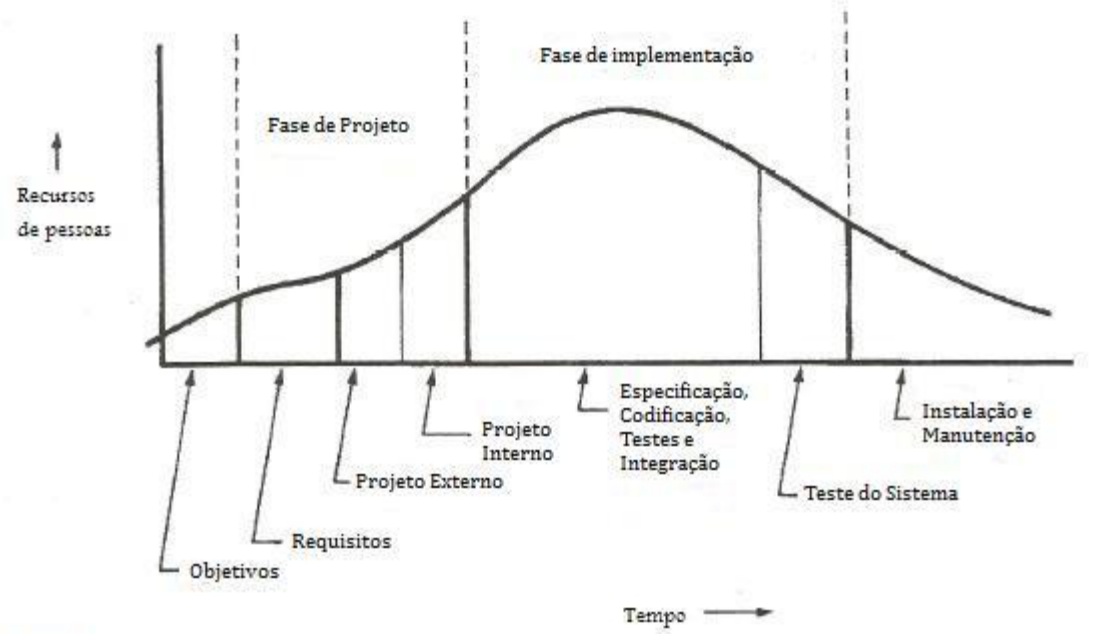


Figura 1

Nossa experiência nos mostra que a fase de projeto utiliza cerca de 20%, enquanto a fase de implementação utiliza cerca de 80% do tempo de trabalho em um projeto completo. Essa distribuição tem sido bastante consistente. Nossos últimos projetos mostraram a mesma distribuição de esforço de trabalho que os primeiros.

Mais tarde, irei mostrar um modelo de porcentagem de esforço de cada tarefa nesse gráfico. Irei mostrar ainda que nós tivemos um aumento de produtividade no projeto durante esse tempo. Uma vez que o formato da distribuição não tenha mudado, a conclusão é que nós obtivemos um aumento de produtividade na fase de projeto do sistema na mesma taxa que na fase de implementação.

As seguintes técnicas de gerência tem nos ajudado a alcançar essa melhoria de produtividade.

Antes que a fase de projeto comece, devemos nos certificar que os objetivos do projeto estão completamente descritos e aprovados, incluindo funções a serem providas, a especificação do trabalho a ser realizado, um cronograma estimado e o custo do projeto.

A fase de concepção do sistema se inicia com a definição dos requisitos, onde os requisitos do negócio são definidos para a aprovação do cliente. Durante o projeto interno e externo do sistema, nós oferecemos ao cliente um esboço do sistema que ele obterá, se for aprovado. Nós fornecemos diagramas e uma proposta de desenvolvimento e implementação. A implementação do sistema segue com um programa de desenvolvimento e termina com um teste e demonstração do sistema para aprovação do cliente. Cada projeto segue essas fases básicas.

Para executar cada projeto, nós usamos um sistema de gerência de projeto (figura 2). A figura 2 nos mostra os processos fundamentais do nosso sistema de gerência de projeto.

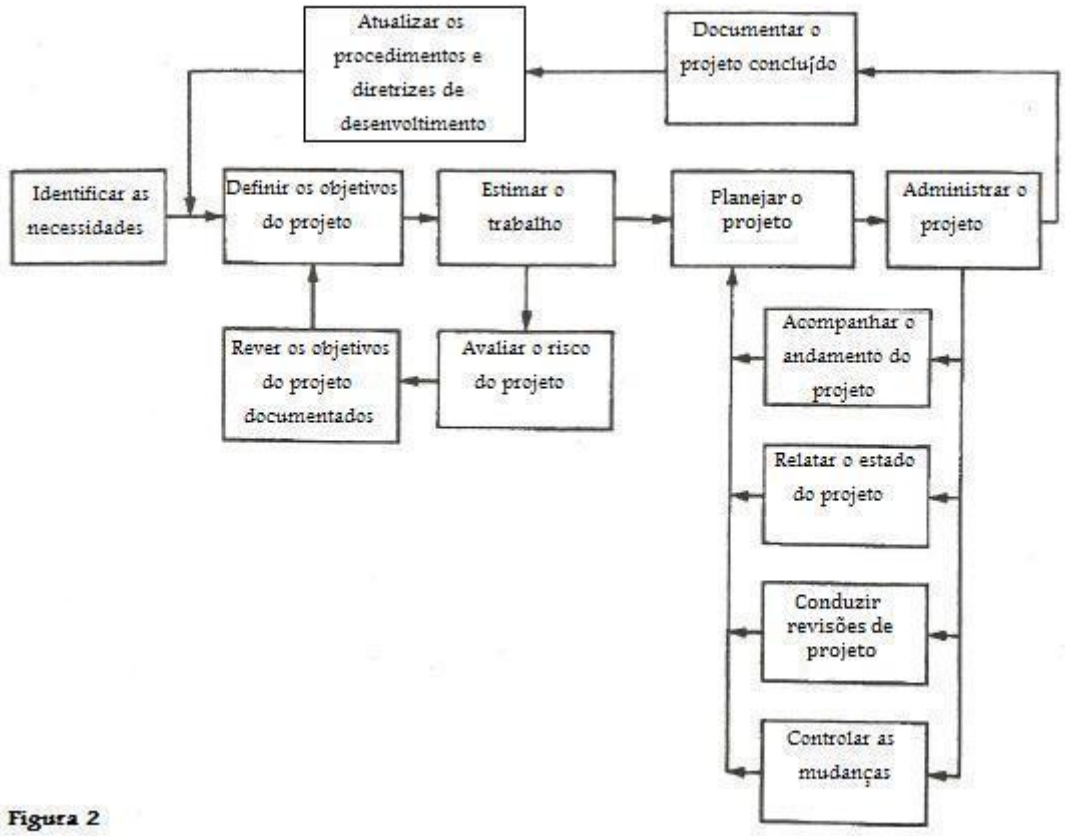


Figura 2

Após identificarmos as necessidades do sistema com o cliente, os objetivos do projeto são completamente documentados. Usando *feedback* de produtividade de projetos anteriores e um cronograma completo das tarefas a serem feitas, nós estimamos o projeto. Em seguida, nós passamos pelo primeiro *loop* de *feedbacks*, onde avaliamos o risco do projeto, com um questionário estruturado em 28 questões. Nós temos um grupo independente de garantia do sistema, que revisa todos os objetivos documentados do projeto, para ter certeza de que está correto, sem ambiguidades e contém todas as informações acordadas. E depois de nós e os clientes entramos em acordo sobre seus objetivos e o planejamento que traçamos para atingi-los, nós começamos o projeto.

O projeto é então feito com detalhes. Nós pedimos que o projeto tenha um tempo médio de 20-40 horas trabalhadas. Nós comparamos os resultados obtidos com o planejamento uma vez por semana, e relatamos o estado do projeto para a gestão da IBM e para o cliente. Nós continuamos as revisões, tendo as garantias do sistema dada a cada seis meses pelo grupo de revisão do projeto. Acima de tudo, nós controlamos as mudanças – não para preveni-las, mas para ter certeza de que cada parte compreendeu o seu valor e o seu custo, e as aprovou, antes de implementá-las.

O elemento chave do projeto de gerência do sistema é: à medida que nós concluímos projetos grandes, com mais de 1000 horas de trabalho, nós documentamos todo o projeto. Quantas horas foram gastas nas diversas tarefas? O que foi entregue? Quais foram as ações úteis ou os eventos prejudiciais? Sob qual sistema e ambiente o desenvolvimento ocorreu? Esse documento é então analisado juntamente com outros relatórios de conclusão de projeto para atualizar os padrões que nós usamos para guiar projetos futuros. Nós medimos a produtividade do desenvolvimento de aplicativos através do *loop* de *feedbacks* no topo da Figura 2, o resultado do projeto completo é documentado.

Agora que eu já falei sobre o nosso processo de desenvolvimento de aplicativos, como nós medimos o seu sucesso? Há três critérios básicos: projetos devem ser finalizados em um tempo pré estabelecido, dentro do orçamento e deve satisfazer ao cliente. O cliente deve especificar desde o início os objetivos funcionais e os objetivos da relação custo/benefício desejados; se o projeto satisfizer esses objetivos, cumprindo o cronograma e o orçamento, o cliente ficará satisfeito.

Afinal de contas, por que medir a produtividade? Devemos medir a produtividade para responder algumas questões. Estamos fazendo o melhor que podemos? Somos competitivos? Estamos nos aperfeiçoando a cada dia? A medição de produtividade é usada para nos dar algumas respostas. Ela nos ajuda a identificar e ressaltar os fatores que melhoram a produtividade e ao mesmo tempo evitar os fatores que a pioram. Nós devemos identificar e selecionar os sistemas de desenvolvimento e tecnologias que oferecem mais funcionalidades de aplicação com menos esforço e com menor custo.

Nós aprendemos que existem algumas coisas a serem levadas em consideração para medir a produtividade. Para começar, você deve avaliar todo o processo, incluindo a fase de projeto. Os custos podem ser avaliados na fase de projeto também. Depois, existem tarefas e funcionalidades como gerencia de projeto e arquitetura de sistema que também devem ser incluídos porque contribuem de forma significativa para o resultado da produtividade. Se ignorarmos esses fatores, teremos uma falsa visão dos custos reais. Finalmente, todas as medições são relativas. Você pode medir projetos atuais entre si, e você pode medir tendências temporais dentro da sua organização. Porém, comparações entre organizações devem ser manuseadas com cuidado, ao menos que elas estejam usando as mesmas definições.

Medição de produtividade pode causar danos. Para evitar isso, mantenha os principais objetivos do projeto em perspectiva – entrega no prazo, não estourar o orçamento, satisfação do cliente. Não deixe que a medição de produtividade desvie sua atenção desses objetivos principais.

Medição de produtividade evita a dependência em medidas como linhas de código, que podem variar de forma considerável, dependendo da tecnologia usada. Nós utilizamos linhas de código, mas apenas como uma medida adicional para comparar projetos que utilizam a mesma linguagem.

Foram analisados projetos de desenvolvimento de 22 aplicações do DP. Desses, 16 foram escritos em COBOL, 4 usaram PL/1, e 2 usaram DMS/VS.

(Apesar de ainda utilizar uma quantidade significativa de código em Assembly *(ALC – Assembly Language Coding)* para funções especializadas em alguns de nossos projetos, nós não fazemos mais projetos inteiros usando ALC, por que não é suficientemente produtivo. Consequentemente, não temos projetos que utilizam ALC em nossas medições. Para aqueles que possam ter os dados, o método descrito deve funcionar bem medindo a produtividade funcional relativa em projetos que utilizam ALC).

Datas de conclusão de projetos variavam de meados de 1974 ao início de 1979. Os projetos variavam em tamanho entre 500 horas de trabalho a 105.000 horas de trabalho. Vinte e dois projetos parece ser o número pequeno, considerando que os serviços do DP provavelmente concluíram 1500 contratos durante esse tempo. No entanto, esses 22 foram todos os projetos que passaram pelo critério de seleção nesse período.

Essas foram as regras dos critérios de seleção:

1- Somente projetos concluídos que passaram por todas as fases, desde a definição dos requisitos até as demonstrações e testes do sistema final e já foram entregues ao cliente, puderam ser analisados.

2- Todo o projeto teve que ser feito sob a nossa gerência de projeto, baseado em definições consistentes e procedimentos de gestão.

3- Todas as horas de trabalho gastas pelos nossos funcionários e pelos clientes devem ser conhecidas e registradas com cuidado.

4- Os fatores funcionais devem ser conhecidos.

Verificamos que cerca de 3 a 7 projetos que cumprem esses critérios são concluídos por ano.

Para medir produtividade, nós tivemos que definir e medir o produto e o custo. O produto analisado, foi o valor das funcionalidades entregues. O número de entradas, consultas, saídas e arquivos mestre entregues foram contados, pesados, somados e ajustados para a complexidade de cada projeto. O objetivo era desenvolver uma medida relativa do valor das funcionalidades entregues ao usuário que fosse independente de uma tecnologia ou abordagem particular utilizada.

A base para esse método foi desenvolvida nos últimos cinco anos pelos serviços de estimativa de projetos do DP. Como parte desse processo de estimativa, nós validamos cada estimativa como uma série de questões ponderadas sobre as funcionalidades da aplicação e o ambiente de desenvolvimento. Verificamos que o valor base da função de aplicação era consistentemente proporcional a contagem ponderada do número de entradas, saídas, consultas e arquivos mestre do usuário externo.

A abordagem geral é contar o número de entradas, saídas, consultas do usuário externo e arquivos mestre entregues pelo projeto desenvolvido. Esses fatores são a manifestação externa de qualquer aplicação, e cobrem todas as suas funcionalidades.

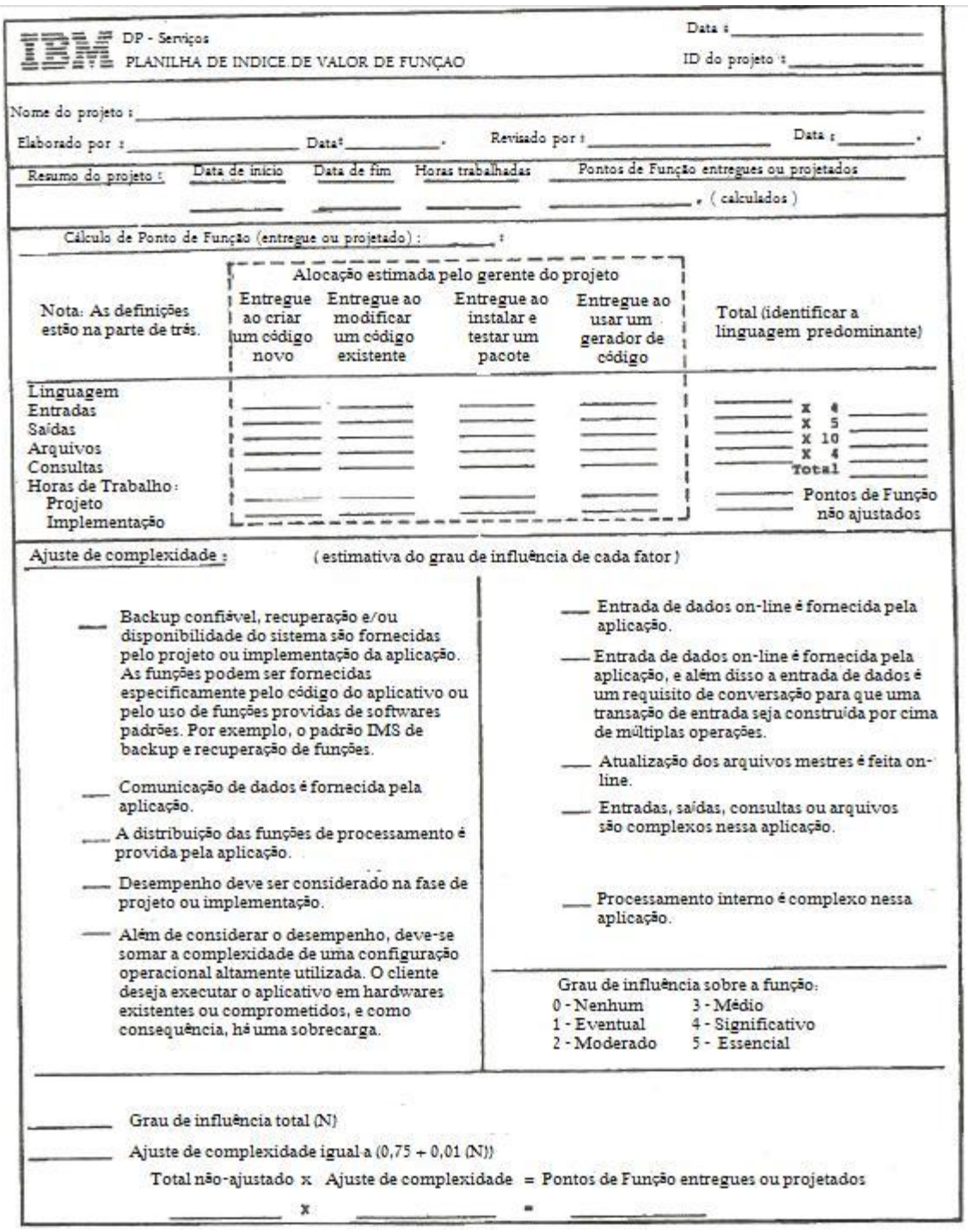
Essas contagens são ponderadas pelos números projetados para refletir o valor da função para o cliente. As ponderações utilizadas foram determinadas por debates e julgamentos. Esses pesos têm nos dado bons resultados:

- Número de Entradas x 4;
- Número de Saídas x 5;
- Número de Consultas x 4;
- Número de Arquivos Mestre x 10;

Então, ajustamos o resultado para o efeito de outros fatores.

Se as entradas, saídas ou arquivos são muito complicados, nós adicionamos 5%. Para Processamento interno complexo, pode-se adicionar mais 5%. Funções *on-line* e performance são abordadas em outras questões. O ajustamento máximo possível é de 50%, expresso por +/- 25%, de modo que a soma ponderada é a complexidade média.

Isso nos dá um número adimensional definido em Pontos de Função, que temos encontrado para ser uma medida eficaz de valor relativo das funções entregues ao cliente. Todas essas definições são mostradas na Planilha de Valor de Função (Figura 3).



DP - Serviços
PLANILHA DE INDICE DE VALOR DE FUNÇAO

Data :
ID do projeto :

Nome do projeto :

Elaborado por : Data : . Revisado por : Data : .

Resumo do projeto : Data de início | Data de fim | Horas trabalhadas | Pontos de Função entregues ou projetados

, (calculados)

Cálculo de Ponto de Função (entregue ou projetado) : :

Nota: As definições estão na parte de trás.	Alocação estimada pelo gerente do projeto: Entregue ao criar um código novo	Entregue ao modificar um código existente	Entregue ao instalar e testar um pacote	Entregue ao usar um gerador de código	Total (identificar a linguagem predominante)
Linguagem					
Entradas					x 4
Saídas					x 5
Arquivos					x 10
Consultas					x 4
Horas de Trabalho:					Total
Projeto					Pontos de Função
Implementação					não ajustados

Ajuste de complexidade : (estimativa do grau de influência de cada fator)

- Backup confiável, recuperação e/ou disponibilidade do sistema são fornecidas pelo projeto ou implementação da aplicação. As funções podem ser fornecidas especificamente pelo código do aplicativo ou pelo uso de funções providas de softwares padrões. Por exemplo, o padrão IMS de backup e recuperação de funções.
- Comunicação de dados é fornecida pela aplicação.
- A distribuição das funções de processamento é provida pela aplicação.
- Desempenho deve ser considerado na fase de projeto ou implementação.
- Além de considerar o desempenho, deve-se somar a complexidade de uma configuração operacional altamente utilizada. O cliente deseja executar o aplicativo em hardwares existentes ou comprometidos, e como consequência, há uma sobrecarga.
- Entrada de dados on-line é fornecida pela aplicação.
- Entrada de dados on-line é fornecida pela aplicação, e além disso a entrada de dados é um requisito de conversação para que uma transação de entrada seja construída por cima de múltiplas operações.
- Atualização dos arquivos mestres é feita on-line.
- Entradas, saídas, consultas ou arquivos são complexos nessa aplicação.
- Processamento interno é complexo nessa aplicação.

Grau de influência sobre a função:
0 - Nenhum 3 - Médio
1 - Eventual 4 - Significativo
2 - Moderado 5 - Essencial

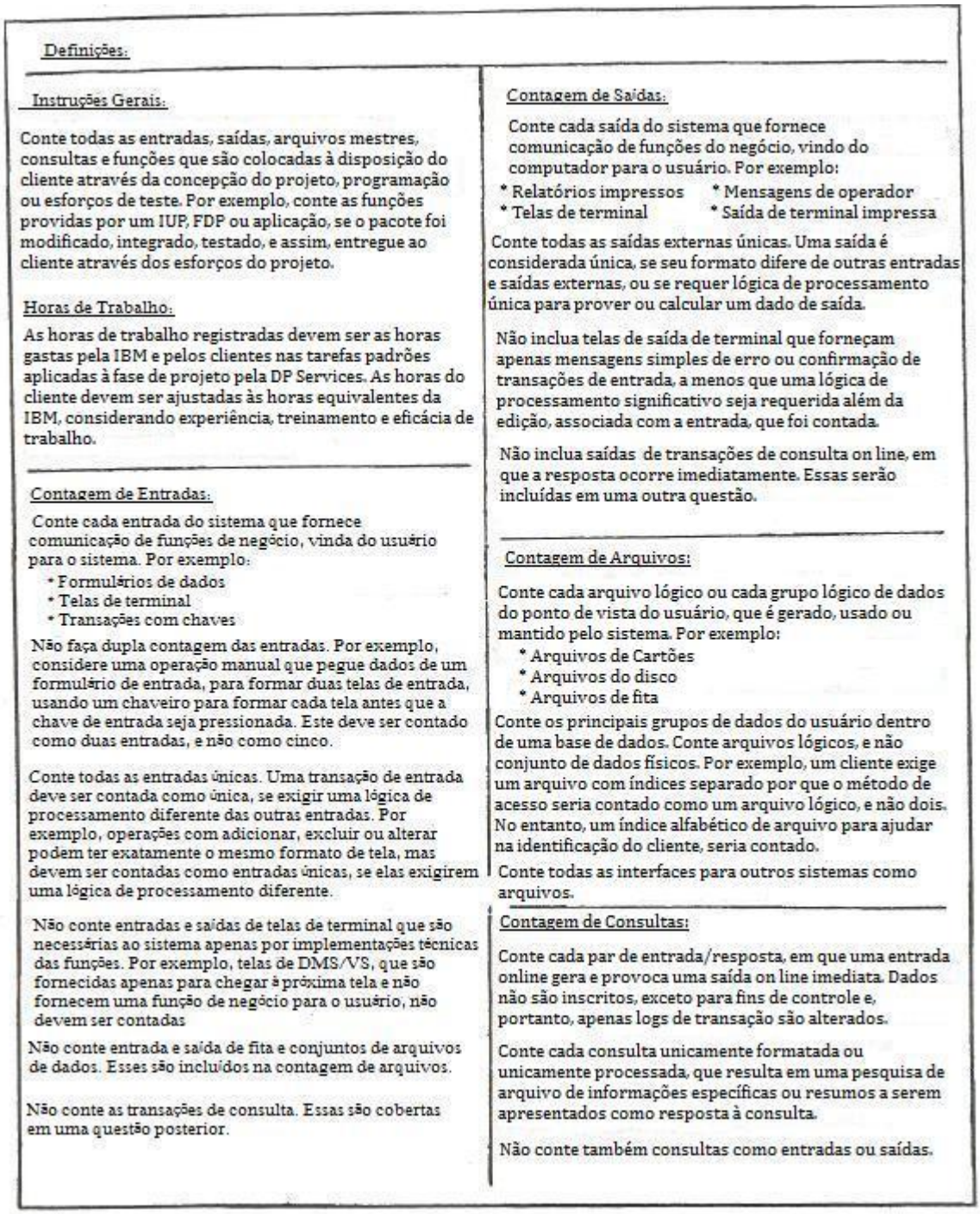
Grau de influência total (N)

Ajuste de complexidade igual a (0,75 + 0,01 (N))

Total não-ajustado x Ajuste de complexidade = Pontos de Função entregues ou projetados

x =

Figura 3



Definições:

Instruções Gerais:

Conte todas as entradas, saídas, arquivos mestres, consultas e funções que são colocadas à disposição do cliente através da concepção do projeto, programação ou esforços de teste. Por exemplo, conte as funções providas por um IUP, FDP ou aplicação, se o pacote foi modificado, integrado, testado, e assim, entregue ao cliente através dos esforços do projeto.

Horas de Trabalho:

As horas de trabalho registradas devem ser as horas gastas pela IBM e pelos clientes nas tarefas padrões aplicadas à fase de projeto pela DP Services. As horas do cliente devem ser ajustadas às horas equivalentes da IBM, considerando experiência, treinamento e eficácia de trabalho.

Contagem de Entradas:

Conte cada entrada do sistema que fornece comunicação de funções de negócio, vinda do usuário para o sistema. Por exemplo:

* Formulários de dados
* Telas de terminal
* Transações com chaves

Não faça dupla contagem das entradas. Por exemplo, considere uma operação manual que pegue dados de um formulário de entrada, para formar duas telas de entrada, usando um chaveiro para formar cada tela antes que a chave de entrada seja pressionada. Este deve ser contado como duas entradas, e não como cinco.

Conte todas as entradas únicas. Uma transação de entrada deve ser contada como única, se exigir uma lógica de processamento diferente das outras entradas. Por exemplo, operações com adicionar, excluir ou alterar podem ter exatamente o mesmo formato de tela, mas devem ser contadas como entradas únicas, se elas exigirem uma lógica de processamento diferente.

Não conte entradas e saídas de telas de terminal que são necessárias ao sistema apenas por implementações técnicas das funções. Por exemplo, telas de DMS/VS, que são fornecidas apenas para chegar à próxima tela e não fornecem uma função de negócio para o usuário, não devem ser contadas

Não conte entrada e saída de fita e conjuntos de arquivos de dados. Esses são incluídos na contagem de arquivos.

Não conte as transações de consulta. Essas são cobertas em uma questão posterior.

Contagem de Saídas:

Conte cada saída do sistema que fornece comunicação de funções do negócio, vindo do computador para o usuário. Por exemplo:

* Relatórios impressos
* Mensagens de operador
* Telas de terminal
* Saída de terminal impressa

Conte todas as saídas externas únicas. Uma saída é considerada única, se seu formato difere de outras entradas e saídas externas, ou se requer lógica de processamento única para prover ou calcular um dado de saída.

Não inclua telas de saída de terminal que forneçam apenas mensagens simples de erro ou confirmação de transações de entrada, a menos que uma lógica de processamento significativo seja requerida além da edição, associada com a entrada, que foi contada.

Não inclua saídas de transações de consulta on line, em que a resposta ocorre imediatamente. Essas serão incluídas em uma outra questão.

Contagem de Arquivos:

Conte cada arquivo lógico ou cada grupo lógico de dados do ponto de vista do usuário, que é gerado, usado ou mantido pelo sistema. Por exemplo:

* Arquivos de Cartões
* Arquivos do disco
* Arquivos de fita

Conte os principais grupos de dados do usuário dentro de uma base de dados. Conte arquivos lógicos, e não conjunto de dados físicos. Por exemplo, um cliente exige um arquivo com índices separado por que o método de acesso seria contado como um arquivo lógico, e não dois. No entanto, um índice alfabético de arquivo para ajudar na identificação do cliente, seria contado.

Conte todas as interfaces para outros sistemas como arquivos.

Contagem de Consultas:

Conte cada par de entrada/resposta, em que uma entrada online gera e provoca uma saída on line imediata. Dados não são inscritos, exceto para fins de controle e, portanto, apenas logs de transação são alterados.

Conte cada consulta unicamente formatada ou unicamente processada, que resulta em uma pesquisa de arquivo de informações específicas ou resumos a serem apresentados como resposta à consulta.

Não conte também consultas como entradas ou saídas.

Figura 3 (Continuação)

Um artigo recente publicado por Trevor Crossman em maio de 1979 descreveu uma abordagem funcional similar para medir a produtividade do programador. A sua abordagem define as funcionalidades baseadas na estrutura do programa. Nossa abordagem define as funcionalidades baseadas em atributos externos. Ele se concentrou em tarefas de projeto do programador, código e teste. Como você verá, nós olhamos todo o ciclo de desenvolvimento da aplicação, desde o sistema de projeto até o sistema de teste. Os nossos pontos de vista sobre a necessidade de uma abordagem funcional parecem estar de acordo.

Eu acredito que a abordagem utilizada pelo DP, baseada em atributos externos, será mais

efetiva para determinar ou prover vantagens de produtividade ou linguagens de nível superior e tecnologias de desenvolvimento.

O custo utilizado foram as horas de trabalho contribuídas tanto pela IBM quanto pelos clientes envolvidos no projeto, durante todas as fases (projeto e implementação) (Figura 4). Esses são os elementos de custo abrangidos na análise. A distribuição percentual no atual modelo tem sido desenvolvida pelos serviços de experiência em projetos do DP nos últimos 3 ou 4 anos. A intenção é indicar o custo do desenvolvimento em termos de horas de trabalho utilizadas nas fases de projeto, programação e teste do desenvolvimento da aplicação.

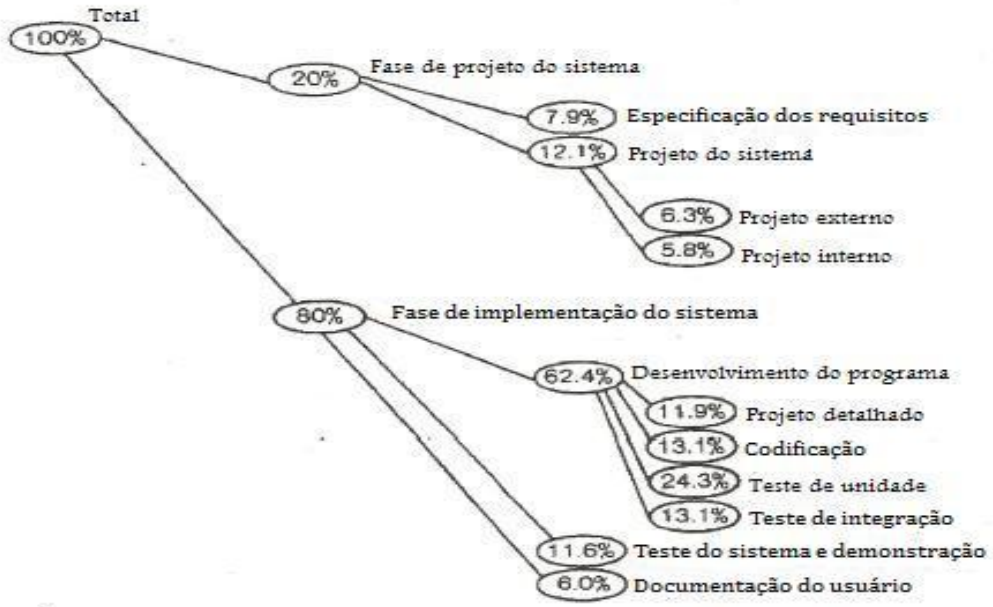


Figura 4

A Figura 5 mostra a nossa tendência temporal de produtividade entre 1974 a 1978. Temos as horas trabalhadas por ponto de função entregues plotadas no gráfico. Isso significa que quanto mais em baixo, melhor; em baixo significa que poucas horas foram gastas gerando cada ponto de função. Existem três tipos de projetos representados aqui. Cada ponto vermelho representa um projeto em COBOL, cada ponto azul representa um projeto em PL/1, e cada ponto branco representa um projeto em DMS/VS. A linha é um ajuste linear dos quadrados mínimos para todos os pontos.

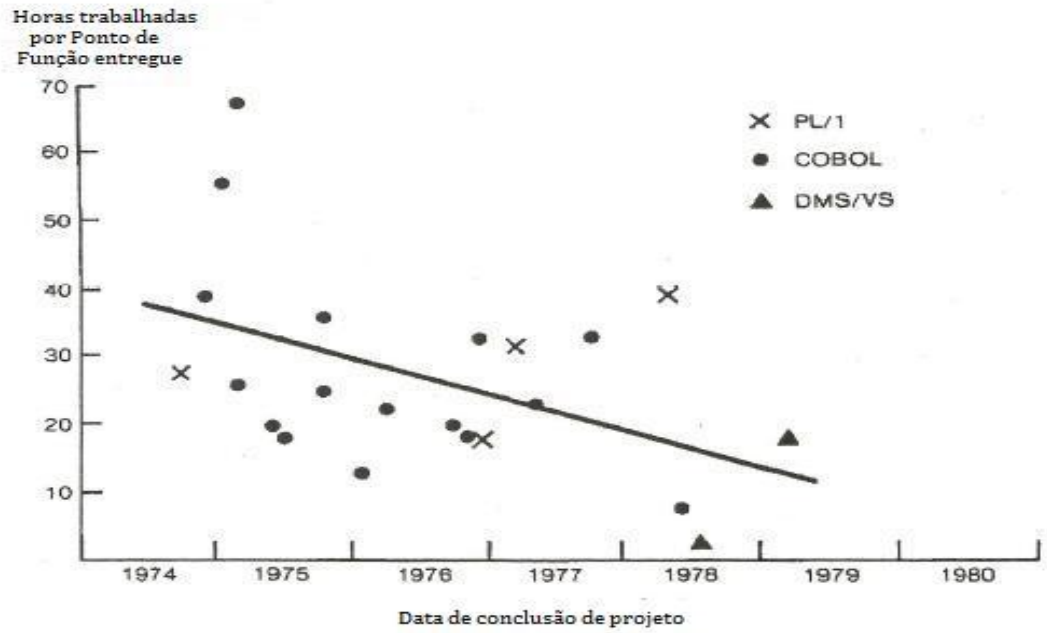


Figura 5

Houve uma melhoria da produtividade de cerca de 3 vezes, mostrada entre 1974 e 1978. Os projetos em PL/1, DMS/VS e COBOL combinados nos mostram uma tendência significativa de aumento de produtividade. Por termos 3 linguagens representadas, nós acreditamos ter uma medida que possa analisar a produtividade relativa de diferentes linguagens e diferentes tecnologias. O projeto com DMS/VS, em particular, tem seguido a tendência do aumento de produtividade, embora estarmos ainda no início da curva de aprendizado com DMS.

A Figura 6 nos mostra o relacionamento entre o tamanho do projeto e a produtividade. Isso é derivado de um mesmo projeto. Isso nos mostra que, à medida que o projeto cresce, mais horas de trabalho são necessárias para produzir cada ponto de função. Já se conhece esse efeito há muito tempo. Esse gráfico simplesmente quantifica o resultado dos 22 projetos que analisamos. Ele nos mostra que o tamanho do projeto deve ser levado em consideração em qualquer comparação de produtividade. Além disso, as duas curvas nos mostra que para toda variedade de tamanhos de projeto, em média, PL/1 é provavelmente cerca de 25% mais produtivo que COBOL. A indicação inicial sobre DMS é que provavelmente seja mais produtiva que PL/1 e COBOL. Se essa tendência é confirmada, DMS/VS poderia revelar ser cerca 30% mais produtiva que COBOL.

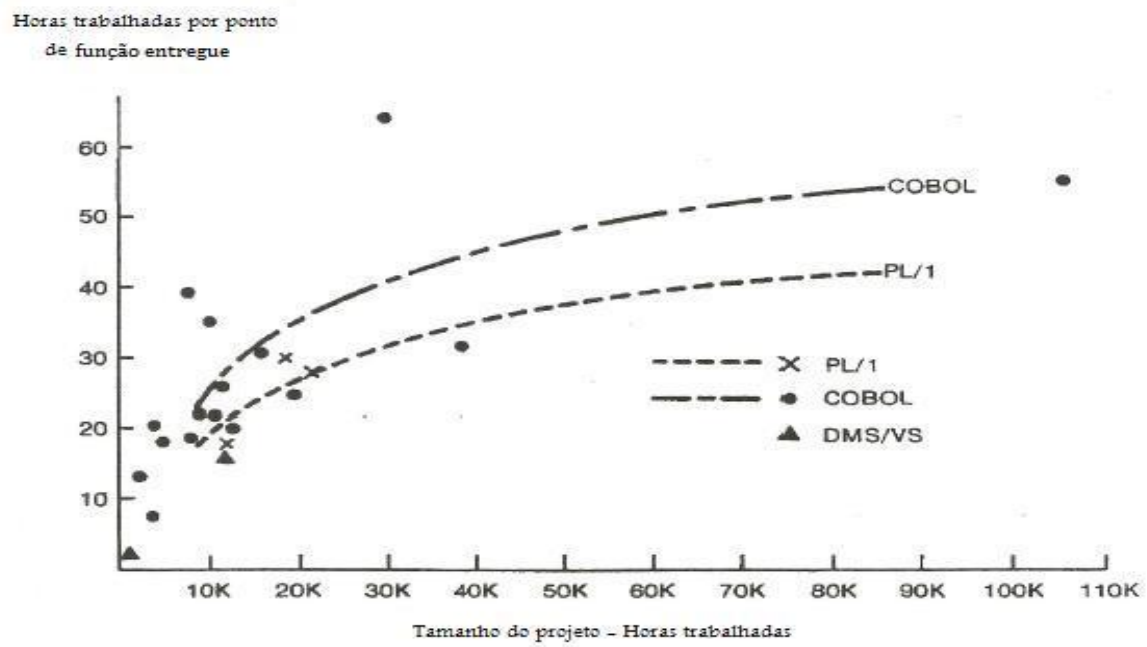


Figura 6

A promessa de ser capaz de analisar a produtividade relativa de técnicas tão diferentes como COBOL e DMS/VS é o maior potencial desta abordagem funcional. Pode fornecer os meios para comprovar a eficiência de linguagens de alto nível e abordagens como DMS/VS e ADF.

A Figura 7 utiliza os mesmos dados, mas elimina o efeito do tamanho do projeto em nossa linha de tendência de produtividade.

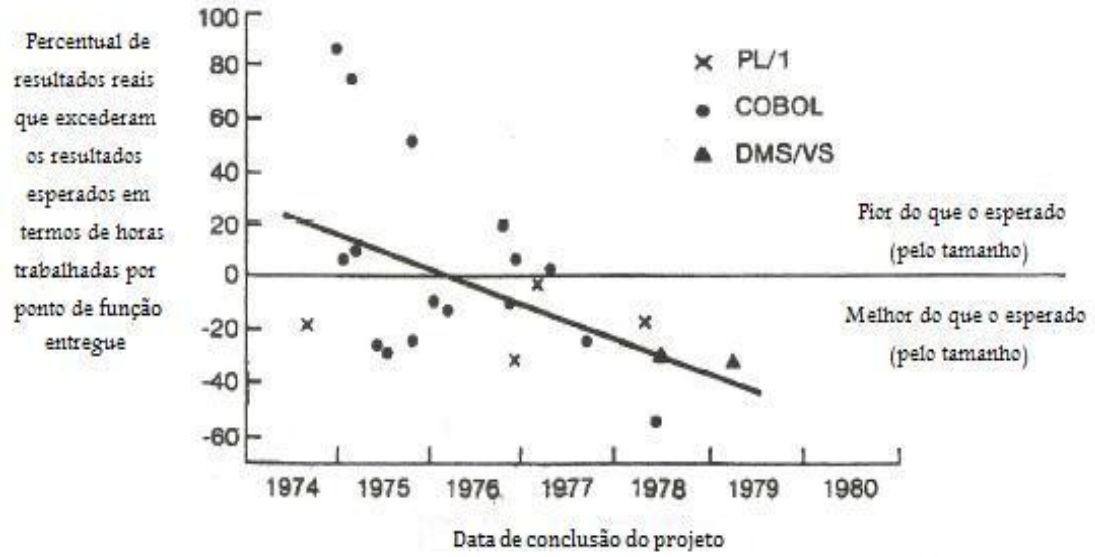


Figura 7

Lembre-se que "baixo" ainda significa que poucas horas foram gastas para produzir cada ponto de função, o que é bom. Com o efeito de tamanho do projeto removido, ainda há uma significante tendência de melhoria de produtividade entre 1974-1978.

A Figura 7 ainda nos permite dividir nosso projeto em dois grupos: Aqueles abaixo da linha zero, que excedeu a expectativa de produtividade; e aqueles acima da linha, que não atingiram a sua expectativa de produtividade. Então podemos tirar algumas conclusões.

Fatores fortemente associados a uma produtividade maior que a média foram:

- O projeto foi concluído depois de 1976, momento em que o processo de desenvolvimento de aplicações disciplinadas (implantadas em 1974) estava começando a ter esse efeito em todos os nossos projetos.
- As linguagens de programação e tecnologias PL/1, DMS/VS e desenvolvimento *on line* foram usados.
- A evolução das tecnologias de programação – código estruturado, implementação *top-down*,desenvolvimento de bibliotecas, e documentação HIPO - foram usadas.

Quando aprovados, esses são agora usados em todos os nossos contratos (naturalmente, decisões como linguagens de programação e o uso de DMS/VS são ainda de direito dos nossos clientes).

O último gráfico nos dá algumas pistas de onde vem essa produtividade. Quantas linhas de código projetadas, escritas e testadas devem ser feitas para produzir um valor de ponto de função? (Figura 8). Para esses projetos, uma média de 25 linhas de código por ponto de função entregues utilizando DMS/VS, uma média de 65 para PL/1, e uma média de 110 linhas de código por um valor de ponto de função em COBOL. Você pode ver o por que DMS/VS e PL/1 estão provando ser mais produtivas que COBOL.

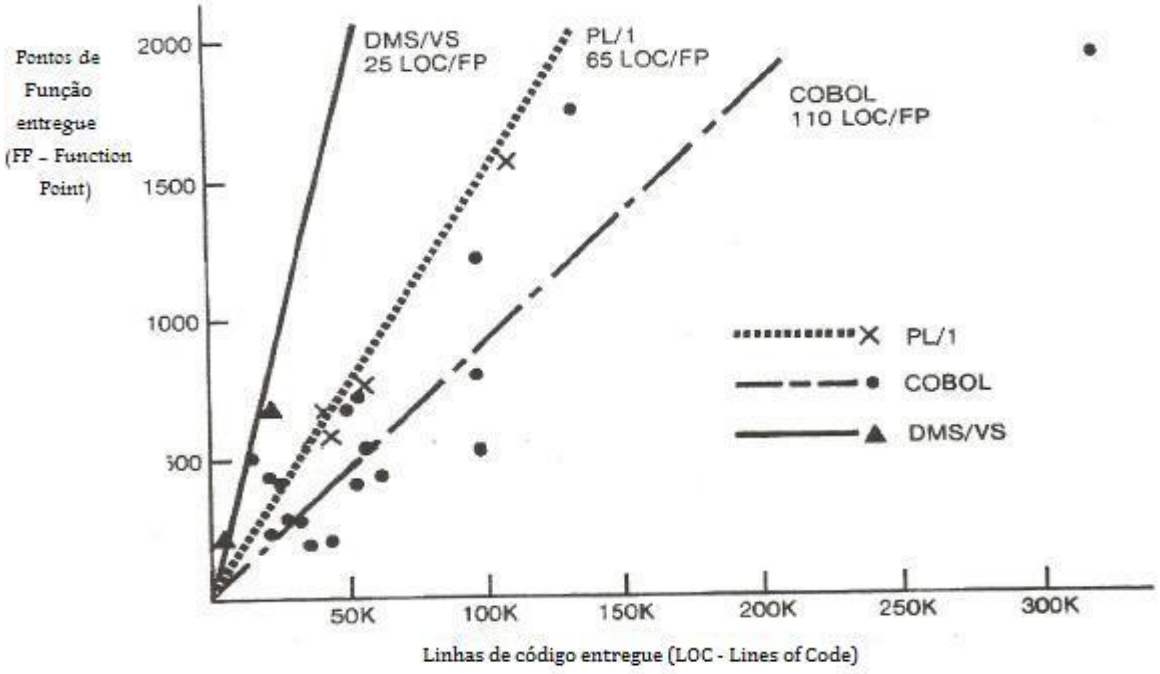


Figura 8

Para resumir, medidas as funcionalidades nos últimos 5 anos, a produtividade do desenvolvimento de aplicações no DP teve um aumento de cerca de 3 vezes. Esse é um aumento de produtividade de cerca de 25% por ano.

Nós temos utilizado a medida do valor da função para determinar a produtividade relativa de diferentes linguagens, tecnologias e tamanhos de projeto. Pretendemos continuar usando essa medida funcional para selecionar e promover tecnologias que ajudam a nos manter competitivos.

Verificou-se que essa melhoria nas tecnologias de programação definitivamente contribuem para a alta produtividade. Elas são uma parte importante em cada um dos nossos contratos.

Encontramos fortes indícios de que o sucesso dos projetos se dá ao processo disciplinado de desenvolvimento de aplicações. O rendimento consistente desses projetos não se dá somente pela utilização de medidas como entrega do produto no prazo estipulado, dentro do orçamento, e satisfação do cliente, como também está fortemente associado com a alta produtividade. Além disso, provê retornos confiáveis de projeto que precisamos para fazer uma análise significativa.

Projetos em fases, que tem sido uma parte importante no nosso processo de desenvolvimento de aplicação, tem nos ajudado a definir nossos projetos em pedaços menores. Por sua vez, esses projetos menores têm contribuído para o aumento da produtividade. Em relação à medição da produtividade dos projetos, verificamos que deve ser incluso todo o processo, incluindo a fase de projeto, na medição de produtividade para representar resultados significativos.

Verificamos que um processo disciplinado tem um ingrediente essencial para o significado das medidas de produtividade. Disciplina fornece definições acordadas de produtos e custos consistentes em projetos que estão sendo medidos.

A medição baseada em funções tem provado ser um meio eficaz de comparar produtividade entre projetos. Antes dessa definição, nós só podíamos comparar projetos de linguagens e tecnologias semelhantes ou senão, tínhamos que encarar a dificuldade em comparar estimativas de projetos hipotéticos e resultados reais. Nós pretendemos continuar usando e aperfeiçoando a medição por valor da função.