UNIVERSIDADE TUIUTI DO PARANÁ

Jason Guilherme Mayer

João Paulo Franqueto

ESTUDO COMPARATIVO ENTRE LINGUAGENS DE PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Curitiba

2005

UNIVERSIDADE TUIUTI DO PARANÁ

Jason Guilherme Mayer

João Paulo Franqueto

ESTUDO COMPARATIVO ENTRE LINGUAGENS DE PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Monografia apresentada ao Curso de Sistemas de Informação da Faculdade de Ciências Exatas da Universidade Tuiuti do Paraná, como requisito para a obtenção do grau em Bacharel em Sistemas de Informação.

Orientador: Evandro Alberto Zatti

Curitiba

2005

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE APÊNDICES

LISTA DE ABREVIAÇÕES

PDA	-	PERSONAL DIGITAL ASSISTANT
API	-	APLICATTION PROGRAMMING INTERFACE
SDK	-	SOFTWARE DEVELOPMENT KIT
MDI	-	MULTIPLE DOCUMENT INTERFACE
DLL	-	DATA LOGIC LIBRARY
VM	-	VIRTUAL MACHINE
ODBC	-	OPEN DATABASE CONNECTIVITY
CLR	-	COMMON LANGUAGE RUNTIME
JIT	-	JUST-IN-TIME
IL	-	INTERMEDIATE LANGUAGE
COM	-	COMPONENT OBJECT MODEL
GUID	-	GLOBAL UNIC IDENTIFIER
MMC	-	MULTIMEDIA MEMORY CARD
SD	-	SECURE DIGITAL
GSM	-	GLOBAL SYSTEM MOBILE
CDMA	-	CODE DIVISION MULTIPLE ACCESS
HP	-	HEWLETT-PACKARD
PHD	-	PHILOSOPHY DOCTOR
IDE	-	INTEGRATED DEVELOPMENT ENVIRONMENT
HTTP	-	HYPERTEXT TRANSFER PROTOCOL
HTML	-	HYPERTEXT MARKUP LANGUAGE
MSDN	-	MICROSOFT DEVELOPER NETWORK
RAM	-	RAMDOM ACCESS MEMORY

1 INTRODUÇÃO

Os assistentes pessoais digitais (PDA - *Personal Digital Assistant*.) tornaram se uma ferramenta indispensável para os executivos. São pequenos computadores de mão, leves e atraentes, que pesam entre 100 e 300 gramas. Para inserir dados, é preciso desenhar as letras com um lápis de plástico em uma pequena tela. O *handheld* reconhece os caracteres gráficos inseridos e os converte em letras e números digitais. Também é possível habilitar um teclado virtual que aparece na tela.

As marcas mais conhecidas são Palm, IBM, Sony, HandSpring, Hewlett Packard e Casio, que possuem um sistema operacional como Palm OS, Windows Mobile, Linux ou Symbian. Estes aparelhos são alimentados com pilhas comuns ou baterias, podendo a maioria deles ser recarregado na corrente elétrica.

Na maioria das vezes são equipamentos com porta infravermelha, *bluetooth* ou *wi-fi* para transmissão de dados sem fio. Os *handheld*s cumprem as funções mais comuns das tradicionais agendas eletrônicas, como lista de telefones, agenda, lista de tarefas, calendários, anotações e calculadora, entre outras. Em alguns casos, contam com versões reduzidas de certas aplicações como Word, Excel, Explorer e Outlook. Também é possível gerenciar fotos, utilizar um MP3 *player*, acessar internet. Um dos grandes benefícios dos dispositivos de mão é a capacidade de ter uma informação em qualquer lugar e depois poder sincronizar tudo com o PC. Outra característica importante é de ter a possibilidade de desenvolver novos aplicativos para as mais diferentes áreas de atuação.

Existe também a possibilidade de ligar periféricos como teclados, mouse, impressoras ou qualquer dispositivo sem fio através da porta infravermelha ou

bluetooth.

A comunidade de desenvolvedores para Palm OS e Windows Mobile, respectivamente sistemas operacionais do Palm e Pocket PC, tem crescido de forma exponencial nos últimos anos.

A criação de aplicativos para dispositivos móveis já foram um grande problema, pela falta de documentação e ferramentas adequadas para desenvolvimento. Atualmente existem várias ferramentas e documentação, facilitando o trabalho de programação para este tipo de equipamento.

O maior problema para iniciar um novo aplicativo para um dispositivo de mão, é a escolha da ferramenta/linguagem de desenvolvimento que pode ser específica para cada sistema operacional, podendo ser escolhido entre o Palm OS, Windows Mobile, Linux e Symbian.

Os sistemas operacionais Palm OS e Windows Mobile são os mais comuns, mas eles contêm muitas características diferentes entre si, como o seu funcionamento lógico e físico. Por serem estes os mais utilizados, a maioria dos softwares desenvolvidos estão disponíveis para estas duas plataformas. As ferramentas de desenvolvimento mais utilizadas para estes sistemas operacionais são o Pocket Studio e Visual Basic .NET.

O objetivo deste projeto é fazer um comparativo entre as linguagens Visual Basic. NET, Pocket Studio e SuperWaba para os sistemas operacionais Windows Mobile e Palm OS, por estes serem os principais sistemas operacionais de dispositivos móveis disponíveis no mercado atualmente. O Visual Basic .NET será usado como ferramenta de desenvolvimento para o sistema operacional Windows Mobile, o Pocket Studio para o Palm e o SuperWaba será usado no desenvolvimento para os dois sistemas operacionais.

Esta pesquisa apresenta a construção de um protótipo exatamente igual para as duas plataformas, usando as três linguagens de programação tratadas neste projeto, com o objetivo de testar o desempenho em cada plataforma.

Após o desenvolvimento do protótipo foi montado um relatório comparativo de cada linguagem, seguindo critérios de comparações de linguagem de programação e um comparativo de desempenho.

2 CRITÉRIOS DE AVALIAÇÃO DE LINGUAGEM DE PROGRAMAÇÃO

Neste capítulo são avaliados os conceitos de várias estruturas e capacidades de programação das linguagens. Esta avaliação será realizada sob o foco do impacto que elas têm sobre o processo de desenvolvimento e manutenção de software. Para isso é necessário que sejam definidos critérios para esta avaliação. Não é fácil definir as características mais importantes, mas podem-se definir as principais características, que de certa forma mais influenciam na classificação das linguagens de programação, conforme o quadro a seguir (SEBESTA, 2003):

QUADRO 1: CRITÉRIOS DE AVALIAÇÃO DA LINGUAGEM - 2003

	Critérios		
Característica	Legibilidade	Capacidade de Escrita	Confiabilidade
Simplicidade/ortogonalidade	X	X	X
Estruturas de controle	X	X	X
Tipos de dados e estruturas	X	X	X
Projeto de sintaxe	X	X	X
Suporte para abstração		X	X
Expressividade		X	X
Verificação de tipos			X
Manipulação de exceções			X
Apelido restrito			X

FONTE: SEBESTA p. 22

2.1 LEGIBILIDADE

Legibilidade pode ser definida como a forma pela qual é escrito o código do programa que possibilita a facilidade de leitura e compreensão do mesmo, e tornou-se um importante parâmetro de qualidade de programas e linguagens de programação. (SEBESTA, 2003).

A legibilidade somente começou a ter importância depois dos anos 70, com o surgimento do conceito de ciclo de vida do software, quando a manutenção de software começou a se tornar constante e o entendimento de um programa já escrito tornou-se essencial. (*ibidem*).

Deve-se levar em consideração também, se um programa foi escrito com uma linguagem apropriada para a atividade que será desenvolvida, pois caso não seja, este fato irá prejudicar a legibilidade. (*ibidem*).

O benefício de uma maior legibilidade num programa é o de deixar mais facilmente a correção de possíveis erros, assim como possíveis modificações de melhoramento. (GERBER, 2000).

2.1.1 Simplicidade Global

Para que uma linguagem tenha uma simplicidade global, é necessário que ela seja de fácil leitura e compreensão, ou seja, a legibilidade do programa, como um fator bem marcante é de que os comandos sejam simples, e sejam fáceis de serem manipulados. (SEBESTA, 2003).

Outro fator muito importante é a quantidade de componentes da linguagem, ou seja, multiplicidade de recursos quanto à quantidade de elementos. Quanto maior for a quantidade de elementos que uma linguagem disponibilize, menor vai ser a simplicidade global, pois se uma linguagem tem uma grande quantidade de elementos, o programador acaba, com isto, escolhendo um subconjunto de elementos para fazer o programa, ignorando os demais. Caso uma outra pessoa tenha um conhecimento de um outro subconjunto diferente do que foi programado, então terá dificuldades para entender o código fonte. (GERBER, 2000).

> Uma Segunda característica que complica uma linguagem de programação é a multiplicidade de recursos – mais de uma maneira de realizar uma operação particular. Por exemplo em C, o usuário pode incrementar uma variável simples de quatro maneiras diferentes:
>
> cont = cont + 1
> cont += 1
> cont++
> ++cont
> (SEBESTA, 2003, p. 23)

No exemplo supracitado, verifica-se quatro formas diferentes de incremento de uma variável inteira na linguagem de programação C. Esta multiplicidade de formas de representação de uma mesma operação pode acabar por confundir o programador (originando erros). Outro exemplo esta no quadro 2, que mostra duas formas de comparação, que podem trazer dificuldades ao programador se estes estivessem disponíveis na mesma linguagem. (GERBER, 2000).

Pode-se ainda mencionar outra característica relevante que influencia na simplicidade de uma linguagem de programação, que é o *overloading*, ou seja, sobrecarga de um operador, dando a um operador (“símbolo”) mais de um significado. Por exemplo: Diga-se que o símbolo de soma (+), numa linguagem sirva para somar elementos inteiros e reais, diga-se que nesta mesma linguagem ele ainda possa ser usado para somar os elementos de matrizes, isto dificulta a leitura por ele se tornar mais confuso, tanto pra quem lê como pra quem escreve o programa. (SEBESTA, 2003).

QUADRO 2: EXEMPLO DE SIMPLICIDADE GLOBAL - 2005

Se (condição…) entao código… Fimse	Compara (condição…) faça código… Fim

2.1.2 Ortogonalidade

Em termos de linguagem de programação, ortogonalidade significa que as instruções primitivas, ou seja, básicas da linguagem, podem ser combinadas, de forma a construir estruturas de controle e estrutura de dados de uma linguagem, mas, principalmente, que, toda combinação de estruturas permitidas por ela é válida e significativa. (SEBESTA, 2003).

Pode-se definir ortogonalidade como uma linguagem de programação com a habilidade de combinar estruturas básicas de uma linguagem de programação, de forma que ela possa gerar através das estruturas básicas um grande número de estruturas mais complexas de dados. (*ibidem*).

Diga-se que numa linguagem de programação tenham-se três tipos de dados: inteiro, caracter, real, e dois tipos de operadores: vetor e ponteiros. Numa linguagem ortogonal, serão possíveis as combinações tanto entre os operadores entre si quanto entre eles e os quatro tipos de dados primitivos da linguagem, permitindo assim a formação de um grande número de estruturas de dados, como mostra o quadro 3. (GERBER, 2000).

Caso não fossem permitidas combinações entre os operadores, diminuiria significativamente a ortogonalidade da linguagem. Assim a falta de ortogonalidade leva à exceção à regra numa linguagem de programação, não permitindo que um ponteiro, por exemplo, seja capaz de apontar para outra estrutura de controle ou de dados. A quantidade de exceções à regra de uma linguagem então é proporcional à ortogonalidade da linguagem. (SEBESTA, 2003).

QUADRO 3: EXEMPLO DE ORTOGONALIDADE - 2005

TipoTeste = estrutura
  Nome: Caracter
  Idade: Inteiro
Fim
	TipoTeste VarTeste

Procedimento exemplo(TipoTeste VarTeste) Inicio código… Fim	

2.1.3 Instruções de Controle

Algumas instruções das linguagens de programação, como por exemplo, o *goto*, reduz em muito a legibilidade do programa, pois tornam difícil manter uma leitura de código de cima para baixo. (GERBER, 2000).

Na década de 70, surgiu a programação estruturada, como uma solução para o problema de legibilidade causada pelas limitações das linguagens de programação das décadas de 50 e 60, sendo que elas possuíam poucas instruções de controle, como por exemplo, a criação da instrução “para” (*for*), para substituição do “*goto*”, como uma instrução de repetição. (*ibidem*).

> Um programa que pode ser lido de cima para baixo é muito mais fácil de entender do que o que exige ao leitor pular uma instrução a outra não-adjacente para seguir a ordem de execução. Em certas linguagens, entretanto, instruções goto que desviam para cima, às vezes, são necessárias; por exemplo, elas constroem laços WHILE em FORTRAN 77. Restringir instruções goto das seguintes maneiras pode tornar os programas mais legíveis:
>
> - Elas devem preceder seus alvos, exceto quando usadas para formar laços.
> - Seus alvos nunca devem estar muito distantes.
> - Seu número deve ser limitado.
>
> (SEBESTA, 2003, p. 26)

O uso das instruções de controle *goto* para fazer laços de repetição só deve ser utilizado caso a linguagem de programação não possua instruções de controle

mais elaboradas. No quadro 4 mostra alguns exemplos de instruções de controle, inclusive o *goto*. (SEBESTA, 2003).

QUADRO 4: EXEMPLO DE INSTRUÇÕES DE CONTROLE - 2005

se (condicao…) entao código… Fimse	para (condicao…) faça código… Fimpara
Ponto GT código… Goto GT	

2.1.4 Tipos de Dados e Estruturas

Um auxílio muito importante para garantir uma boa legibilidade numa linguagem de programação são os tipos de dados. Com eles pode-se melhorar significativamente a facilidade de leitura e compreensão de um programa. Por exemplo: diga-se que em uma linguagem de programação não tenha o tipo lógico (boleano) e que nela seja usado um tipo numérico para representá-lo.

Nesta linguagem teria - se:

Atribuicao_da_variavel=1

Neste caso, o significado, do que se está querendo dizer não fica bem claro. No entanto, se esta linguagem possuísse o tipo de dados lógico, então ela ficaria assim:

Atribuicao_da_variavel=verdadeiro

Ou seja, a atribuição da variável é falsa ou verdadeira, e seu significado fica perfeitamente compreensível. Um outro exemplo está disponível no quadro 5. (SEBESTA, 2003).

QUADRO 5: EXEMPLO DE TIPOS DE DADOS E ESTRUTURAS - 2005

Casado = 1 ou Casado = 0	Casado = verdadeiro ou Casado = falso

2.1.5 Considerações Sobre a Sintaxe

A sintaxe das linguagens de programação, ou seja, a forma dos elementos da linguagem tem um efeito significativo na legibilidade dos programas. (GERBER, 2000).

As limitações nos nomes de variáveis, como ocorriam nas primeiras linguagens de programação, impedem que o nome definido expresse com clareza a função da variável. (SEBESTA, 2003).

Ao formar grupos de códigos em algumas linguagens, que normalmente iniciam em um *for,* while ou *if* podem não ser claros, pois a instrução que indica a finalização do bloco, não indica qual das instruções iniciadas está sendo finalizada. No quadro abaixo é possível verificar um exemplo de blocos, usando a linguagem Pascal, que não utiliza um bom finalizador de blocos e usando também a linguagem Basic que utiliza um identificar melhor para fechar o bloco. (*ibidem*).

QUADRO 6: COMPARAÇÃO DA SINTAXE DO PASCAL E BASIC - 2005

Pascal	**Basic**

if (Condição) then
begin
(instruções...);
end;
	If (Condição) then
	(instruções...)
	endif

Outro item importante é que as linguagens devem impossibilitar o uso de nomes de variáveis com os nomes reservados para as instruções de controle, evitando assim que o entendimento do código fique mais difícil. (GERBER, 2000).

2.2 CAPACIDADE DE ESCRITA

Esta é a característica de como uma linguagem pode ter uma boa capacidade de escrita para o problema a ser solucionado, mas sem perder a

legibilidade, ou seja, permitindo uma fácil leitura por outros programadores. (SEBESTA, 2003).

A comparação entre duas linguagens de programação deve ser feita entre linguagens que tem um mesmo domínio da aplicação a qual foi desenvolvida. Sendo assim seria injusta a comparação entre uma linguagem que foi projetada para certo tipo de aplicação e a outra não. (*ibidem*).

A seguir, os fatores mais importantes que influenciam a capacidade de escrita:

2.2.1 Simplicidade e Ortogonalidade

Quando uma linguagem tem várias formas de construção, pode ocorrer o caso de desuso de recursos que deixam o código menos elaborado e até ineficiente se comparado a melhor utilização dos recursos da linguagem. Isto ocorre quando o programador não tem um domínio amplo da linguagem com que trabalha. Um exemplo está no quadro 7, onde mostra o exemplo de um comando de comparação, que caso possa ser realizado de duas maneiras na mesma linguagem vai prejudicar a simplicidade e ortogonalidade. (SEBESTA, 2003).

Utilizar um número mínimo de recursos primitivos e uma boa forma de combiná-las se mostra benéfico, pois deixa uma melhor ortogonalidade. Assim será mais fácil resolver problemas complexos de forma simples, sem prejudicar a ortogonalidade. Mas o uso exagerado pode comprometer a capacidade de escrita, já que alguns erros de código podem passar despercebidos pelo compilador. (GERBER 2000).

QUADRO 7: EXEMPLO DE SIMPLICIDADE DE ORTOGONALIDADE - 2005

se (condição…) entao
  código…
fimse
	compara (condição…) faça
	  código…
	fim

2.2.2 Suporte a Abstração

Abstração permite a definição e o uso de estruturas e operações complexas de modo que se possa ignorar seus detalhes de construção. É essencial para que se possa tratar problemas de programação complexos, refletindo o modo como resolvemos nossos problemas do dia-a-dia. (SEBESTA, 2003).

Abstrações podem estar relacionadas a processos (funções, procedimentos e outros). Também podem estar relacionadas a dados (estruturas, registros e outros). Veja o exemplo de função e procedimento no quadro 8. (*ibidem*).

Em orientação a objetos, um objeto é uma abstração tanto para dados (atributos, conhecimento de um objeto) quanto para processos (métodos que implementam mensagens são as responsabilidades de um objeto). (*ibidem*).

O uso de abstrações também afeta a legibilidade de um programa, já que simplesmente identificar uma abstração do que observar todos os seus detalhes, facilita a leitura. (*ibidem*).

QUADRO 8: EXEMPLO DE SUPORTE A ABSTRAÇÃO - 2005

Procedimento Teste(Inteiro Parametro) Inicio   código… fim

Funcao Teste(Inteiro Parametro): Logico
Inicio
  código…
  Resultado Verdadeiro
ou
  Resultado Falso
fim

2.2.3 Expressividade

Expressividade pode se referir diferente a características. Por exemplo, na Linguagem C utilizar **cont++** é mais elegante do que se usar **cont = cont + 1**, como em Linguagem Pascal é melhor usar **inc(cont)** do que **cont := cont + 1**. A inclusão da instrução *for* em uma linguagem de programação torna mais fácil à escrita de laços de contagem do que com o uso de *while*, por exemplo. (GERBER, 2000).

Todas estas características aumentam a capacidade de escrita de uma linguagem. Para exemplificar isto, pode ser verificado o quadro 9, onde mostra 2 formas de fazer laços de repetição, sendo o para mais elaborado para ser usado quando se conhece o valor limite de execução, no caso 10 vezes. (*ibidem*).

QUADRO 9: EXEMPLO DE EXPRESSIVIDADE - 2005

I = 0 enquanto (I < 10) Inicio código… I++ fimenquanto	para I = 1 até 10 faca Inicio código… Fimpara

2.3 CONFIABILIDADE

Um programa é confiável quando ele funciona conforme suas especificações sem nenhum tipo de anormalidade. Os sub-tópicos a seguir informam como uma linguagem pode influenciar na confiabilidade do programa por ela gerada. (GERBER, 2000).

2.3.1 Verificação de Tipos

A verificação de tipos consiste em testar a existência de erros durante a compilação evitando que o erro apareça na execução do programa. Sendo a opção de verificação durante a compilação melhor, já que quanto antes for detectado o erro no programa, menos tempo será necessário para arrumá-lo. Caso o erro não seja encontrado, poderá ocorrer durante a utilização do programa pelo usuário, ficando ainda mais difícil diagnosticar a causa do problema. (SEBESTA, 2003).

Um exemplo é a utilização do tipo indeterminado de variável, que é o *variant*. Com esta utilização será possível atribuir qualquer valor para a variável, seja ele texto, inteiro ou real. Caso aconteça uma situação em que a variável seja preenchida com um texto e em seguida tente-se dividir por um número qualquer, ocorrerá um erro pois não é possível dividir o texto. Sendo assim, vai ocorrer um erro durante a execução do programa, pois o compilador não vai conseguir determinar este erro durante a compilação. Neste exemplo, a verificação foi feita, porém durante a execução. Se este problema ocorrer de forma aleatória, será difícil simular o erro e resolver o problema. Mas caso a variável seja de um tipo inteiro e esteja tentando atribuir um conteúdo texto para esta variável, vai ocorrer um erro no momento da compilação. Veja um exemplo no quadro 10. (GERBER, 2000).

Outro exemplo que pode ocorrer é utilizando a Linguagem C ANSI, quando o tipo de um argumento em uma chamada de função não é verificado para verificar se o mesmo coincide com o parâmetro da função. Nesta situação, tanto durante a compilação quanto durante a execução, não será verificado quando ao dar entrada em um número real deveria ser passado um número inteiro, podendo ocorrer problemas de difícil diagnóstico. (SEBESTA, 2003).

QUADRO 10: EXEMPLO DE VERIFICAÇÃO DE TIPOS - 2005

Inteiro Quantidade Caracter Nome Quantidade = ‘Teste’ Nome = 1	Inteiro Quantidade Caracter Nome Quantidade = 1 Nome = ‘Teste’

2.3.2 Manipulação de Exceções

Poder interceptar um erro em tempo de execução e depois aplicar medidas para que o erro não comprometa o funcionamento do programa, é um grande auxílio para manter a confiabilidade do mesmo. Linguagens atuais como C++, Java e Pascal possuem recursos para tratamento de exceções, mas existem linguagens como C e Fortran, que quase não possuem recursos para tratamento de exceções. A falta deste recurso pode comprometer todo o programa, caso algum erro aconteça, pois o sistema provavelmente será fechado ou ficará travado. Veja no quadro abaixo 2 exemplos de código que podem gerar exceções, sendo o primeiro por estouro de variável e o segundo por divisão por zero. (GERBER, 2000).

QUADRO 11: EXEMPLO MANIPULAÇÃO DE EXCEÇÕES - 2005

Inteiro I I = 15512142124212697254256	Real I I = 15/0

2.3.3 Apelido (*aliasing*)

Apelido é ter dois ou mais nomes fazendo referência à mesma célula de memória. Este é um recurso perigoso, pois ao alterar o valor de uma variável, será alterado o valor da outra variável que faz referência a ela, podendo causar dificuldades para entender o código e procurar soluções, caso algum problema esteja ocorrendo. Ao definir duas variáveis do tipo ponteiro, que apontem para o mesmo local de memória está ocorrendo Apelido, já que ao alterar o conteúdo de

um endereço de memória, o outro ponteiro vai enxergar o novo valor e caso isto não seja previsto pelo programador, poderá trazer problemas. Isto é um efeito colateral da natureza dos ponteiros e das passagens de parâmetros por referência, onde ocorre um caso semelhante, como mostra o quadro a seguir. (SEBESTA, 2003).

QUADRO 12: EXEMPLO DE APELIDO (ALIASING) - 2005

Procedimento Teste(Inteiro Valor) Inicio Valor = 15 fim
V = 1 Teste(V) Imprima(V)

2.3.4 Legibilidade e Capacidade de Escrita

A legibilidade e capacidade de escrita influenciam diretamente na confiabilidade, pois uma linguagem que não possibilite expressar de forma clara e natural os algoritmos tem mais chances de estarem errados. Quanto mais fácil for escrever um programa, mais garantida será sua eficiência. (SEBESTA, 2003).

A legibilidade afeta a confiabilidade tanto no momento de desenvolvimento do programa, quando no período de manutenção, já que programas de difícil leitura vão prejudicar sua alteração. (*ibidem*).

2.4 CUSTO

O custo final de uma linguagem de programação pode ser influenciado por vários fatores, como custo do treinamento de programadores, custo de escrita, compilação, execução, sistema de implementação da linguagem, custo da má confiabilidade e de manutenção dos programas. (SEBESTA, 2003).

2.4.1 Custo do Treinamento de Programadores

Este treinamento é influenciado pela simplicidade da linguagem e pelas experiências dos programadores, embora linguagens mais poderosas não são, necessariamente, sejam difíceis de aprender, mas normalmente são. (SEBESTA, 2003).

2.4.2 Custo de Escrita

Este custo está ligado com a capacidade de escrita da linguagem, cujos esforços para criar programas usando linguagens de alto nível tiveram como objetivo diminuir o custo de produção do software, já que linguagens de alto nível oferecem uma capacidade de escrita melhor. (SEBESTA, 2003).

2.4.3 Custo de Compilação

A compilação do programa depende do computador e do compilador. Se o compilador for muito pesado e precisar de um computador de alto desempenho, logo o custo de compilação será maior, já que o computador vai custar mais caro. Claro que isto, não é um grande problema atualmente, pois os computadores já possuem preços mais acessíveis e de alto desempenho, ao contrário de alguns anos atrás, onde algumas linguagens necessitavam de um *mainframe* somente para esta finalidade. (SEBESTA, 2003).

> Um grande empecilho para os primeiros usos da Ada era o custo proibitivamente elevado para os compiladores Ada de primeira geração. Este problema foi diminuído pelo surgimento de compiladores melhores. (SEBESTA, 2003, p. 31)

2.4.4 Custo de Execução

Este custo é influenciado diretamente pelo projeto da linguagem de programação, sendo que a linguagem deve proporcionar uma execução de código rápido, que irá influenciar no desempenho do software criado. Isto deve ser considerado em uma linguagem que compile programas para executar em dispositivos móveis, já que os mesmos têm um desempenho muito inferior aos computadores tradicionais. (SEBESTA, 2003).

2.4.5 Custo do Sistema e da Implementação da Linguagem

Quando uma linguagem exige hardware e sistemas operacionais caros para executar, certamente terá uma aceitação menor. (SEBESTA, 2003).

2.4.6 Custo da Má Confiabilidade

Se o software falhar em um sistema crítico, como uma usina nuclear, em virtude de uma linguagem, o custo poderá ser muito elevado. Em sistemas não críticos também podem sair muito caros em termos comerciais e em termos jurídicos. (SEBESTA, 2003).

2.4.7 Custo de Manutenção dos Programas

Inclui tanto correções como modificações, pois a evolução de um software depende diretamente da linguagem em que foi criado, já que novos recursos na linguagem podem melhorar a legibilidade em certos casos. Em linguagens que não oferecem boa legibilidade, a manutenção do software pode ser algo desafiador. (SEBESTA, 2003).

3 COMPILAÇÃO DO CÓDIGO FONTE

A compilação do código fonte faz a conversão do código digitado pelo programador em *object code*, que é a forma que o computador vai usar para executar o programa compilado. (AHO, SETHI, ULMAN, 1995).

3.1 LINGUAGENS COMPILADAS

Um programa compilado nada mais é do que um programa escrito em uma linguagem de programação que após a compilação (cada linguagem de programação possuiu um compilador específico) gera uma seqüência de instruções que será executado pelo sistema operacional. Este código compilado é chamado de *object code*. Cada vez que um programa é compilado, ele é compilado especificamente para um sistema operacional. O programa compilado tem a vantagem de ter um desempenho superior a outros tipos de compilação, pois o código é transformado em código de máquina, que será executado diretamente pelo processador do computador. Alguns exemplos de linguagens compiladas são: C, Pascal, Basic, entre outros. (*ibidem*).

3.2 LINGUAGENS INTERPRETADAS

Interpretadores são programas que lêem um código-fonte de uma linguagem de programação e os convertem em código executável. Seu funcionamento pode variar de acordo com a implementação. Em muitos casos o interpretador lê linha-a-linha e converte em *object code* a medida que vai executando o programa. Linguagens interpretadas são mais dinâmicas por não precisarem compilar, o que facilita a atualização e manutenção. Por ser interpretada no da execução, seu

desempenho é inferior a uma linguagem compilada. Alguns exemplos são: JavaScript, PHP, Perl, entre outros. (*ibidem*).

3.3 MÁQUINAS VIRTUAIS (SISTEMA HÍBRIDO)

Uma máquina virtual cria um código intermediário entre o programa que vai ser executado e o sistema operacional. Assim, um programa compilado para uma máquina virtual gera o *byte code*, que pode ser executado em vários sistemas operacionais, desde que possuam uma máquina virtual compatível instalada.

Por ser executado com uma máquina virtual o programa gerado precisa ser “emulado” pela máquina virtual para executar e com isto gera uma perda de desempenho considerável se comparar-se com um programa compilado nativamente para um determinado sistema operacional. Em contra partida deste método é possível compilar um programa e executá-lo em qualquer sistema operacional desde que esteja com uma máquina virtual compatível instalada. Um exemplo de máquina virtual e Java e o recém criado. NET framework da Microsoft. (*ibidem*).

4 DISPOSITIVOS MÓVEIS

Atualmente, várias profissões estão precisando de informações atualizadas e disponíveis em qualquer lugar. Esta informação pode estar disponível em papel ou de forma eletrônica num computador. Mas o importante é manter esta informação acessível, e que não necessite de um deslocamento físico para recebê-la. Para resolver este problema de deslocamento, o Assistente Pessoal Digital (PDA), também conhecido como *handheld*, está se tornando muito utilizado, devido sua funcionabilidade, portabilidade e facilidade de uso. (PALM LAND, 2005).

O PDA é o termo usado para designar qualquer dispositivo pequeno e móvel com capacidade computacional e de armazenamento, sendo ele de uso profissional ou pessoal. (*ibidem*).

Como entrada de dados normalmente é utilizada uma caneta ao invés de um teclado, como ocorre em um computador normal. Esta caneta é utilizada diretamente na tela do *handheld*, que possui um sistema de reconhecimento de toque.

Como ocorre nos computadores normais, o *handheld* necessita de um sistema operacional e de um software para fazer o controle e armazenamento das informações. Estas informações podem ser sincronizadas com um computador, ou seja, tudo o que foi atualizado via *handheld* pode ser transmitido para o computador ou vice-versa. (*ibidem*).

Dois grupos de *handheld* dominam o mercado atualmente. Os baseados no sistema operacional PalmOS e os baseados no sistema operacional Windows CE, da Microsoft, conhecido como Windows Mobile. (*ibidem*).

4.1 PALM

O nome Palm hoje virou símbolo de tecnologia móvel, a exemplo da *discman* e *walkman* da Sony, hoje em dia é comum que ao se dizer que tem um computador de mão diga-se que tem um *Palm*, sendo uma clássica associação de nome ao tipo do produto, como Bombril, relacionado com esponja de aço. (PALM LAND, 2005).

Os Palms, que são todos os PDA cujo sistema operacional é PalmOS, são fabricados por diversos fabricantes de dispositivos móveis, tais como a Sony, com sua linha Clié e a Handspring, que recentemente foi comprada pela PalmOne. Veja exemplos de Palms na figura a seguir. (PALM LAND, 2005).

FIGURA 1: Palm Tungsten T5 E Sony Clié Peg-Ux40 - 2005
FONTE: PalmLand. Disponível em: http://www.palmland.com.br.

4.1.1 História

A Palm Computing foi fundada em 1992 e em 1995 foi comprada pela U.S. Robotics Corp.. Em 1996, a Palm apresentou o Pilot 1000 e o Pilot 5000, que levaram ao ressurgimento dos computadores de mão. Em junho de 1997, Palm se tornou uma subseção da 3Com Corp., pois a U.S. Robotics foi comprada pela 3Com. (PALM BRASIL, 2005).

A Palm acrescentou capacidade de comunicações sem fios avançadas à plataforma do Palm OS entrando no mercado de dispositivos móveis, com telefones celulares, dispositivos de mensagens, comunicadores de dados e *smartphones*, com a aquisição do Smartcode Technologie em fevereiro de 1999. Em setembro de 1999, 3Com anunciou publicamente planos para tornar a Palm Computing em uma empresa comercial independente, o que veio a ocorrer em 2 de março de 2000. (*ibidem*).

Em outubro de 2003, a Palm Computing foi dividida em duas empresas, uma com o nome de PalmSource, que passou a se preocupar apenas com a parte de software e sistema operacional PalmOS e a outra com o nome de PalmOne que passou apenas a cuidar da parte de hardware. A companhia que foi então formada é conhecida como PalmOne. Esta empresa se tornou um líder do novo mercado e tornou-se a empresa mais forte em computadores de mão e hardware de comunicações e soluções de software móvel. (PALMONE, 2005).

Os PDA da Palm mantêm compromissos, contatos, tarefas, possui características e capacidades de multimídia como músicas, vídeos e fotografias digitais. Também existe a possibilidade de criar e editar arquivos compatíveis com Word, Excel e PowerPoint e depois sincroniza tudo com seu computador. (*ibidem*).

Existe uma linha de PDA da Palm que integram as características do Palm com um telefone celular. Estes aparelhos são conhecidos por *smartphones* e são compatíveis com ambas as tecnologias atuais de telefonia, como GSM e CDMA, dando aos consumidores uma ampla opção de escolha. (*ibidem*).

A maioria dos *handhelds* da PalmOne também vem com a capacidade de acessar e administrar seu e-mail pessoal e corporativo usando o software VersaMail,

que suporta clientes de e-mail populares como Microsoft Outlook, Microsoft Outlook Express, Lotus Notes, e Eudora, assim como contas IMAP e POP. (*ibidem*).

4.1.2 Recursos

Basicamente os *palmtops* possuem uma estrutura muito parecida entre eles. Eles possuem 4 botões na parte inferior, abaixo da tela de cristal líquido as quais permitem o acesso as principais funções do *Palmtop*, que são totalmente configuráveis para qualquer aplicativo que esteja instalado, o padrão é Agenda de Endereços, Agenda de Compromissos, Lista de Tarefas e Memorandos, podendo variar em cada modelo. (GRUPO PALM BR, 2005)

No centro entre os botões, existe um direcional, que se pode considerar como setas direcionais, que permite que se tenha acesso aos aplicativos sem o uso da caneta. Logo acima dos botões, e junto com a tela para os modelos com resolução de vídeo de 320X320 pixels ou inferiores, existe uma área de escrita especial que é reservada para o reconhecimento da escrita. Para os modelos que tem a tela maior como a resolução de vídeo de 320X480 pixels, não têm essa área física para a escrita, mas isso é feito por software, que mostra a área de reconhecimento de escrita na parte inferior da tela. Nos cantos dessa área de reconhecimento existem também botões que permitem acessar as funções: *home*, menu, sincronismo e a função localizar. Todos esses botões são acessados através da caneta, usada para o toque na tela. (*ibidem*).

Em quase todos os modelos existe uma porta infravermelho que permite a troca de informações entre *palmtops*, ou ainda entre o *palmtop* e outros dispositivos como: PC, impressoras e teclados. Alguns modelos mais modernos, que já possuem

tecnologias mais avançadas como *bluetooth* ou *wireless*, para comunicação sem fio entre os dispositivos. (*ibidem*).

Os *palmtops* até a versão 4.x do PalmOS vinham equipados com processadores Motorola que variavam de 8MHz a 66MHz de clock. Já os novos *palmtops*, mais modernos que vêm possuem o sistema operacional PalmOS 5 ou superior vem equipados com processadores padrão ARM, que podem ser da Intel, Texas Instruments, ou outras empresas, e seu poder de processamento varia de 144MHz a 400MHz de clock,(ou superior). (*ibidem*).

Quanto ao quesito memória, o Palm pode ter dois tipos de memória: memória RAM, e a memória *flash*. A memória RAM é utilizada para guardar os programas e outras informações, que se tem no *palmtop*, esta memória é considerada memória volátil, ou seja, ela tem que ter uma carga de energia que mantenha os dados na memória, no caso de uma falta de energia os dados serão perdidos, ou seja, se acabar a bateria do *Palm*, ela perde seu conteúdo, ela funciona da mesma forma que a memória RAM dos PCs, (PALM LAND, 2005).

A memória *flash* é onde é armazenado o sistema operacional do Palm. Essa memória, diferentemente da memória RAM, não depende de energia para manter seu conteúdo, sendo assim, mesmo que o *Palm* fique muito tempo sem carga, às informações não se perdem. Neste caso é essencial para que não seja perdido o sistema operacional do dispositivo. (ibidem)

A tela do Palm pode variar muito, pois a Palm, mantém linhas de baixo custo com tela preto e branco, e modelos de alta resolução com telas coloridas. O tamanho da tela pode variar de 160X160 pixels atualmente com o modelo Zire 21 e o Zire 31, até 320X480 pixels com o modelo do Tungsten T5. (PALM BRASIL, 2005).

4.2 POCKET PC

Pocket PC é uma denominação dada a todos os *handheld* que utilizam como sistema operacional o Windows CE executando o *shell* Pocket PC. Existem vários fabricantes de Pocket PC, entre eles, os mais conhecidos são HP com a sua linha IPaq, a Dell com a linha Axim, entre outros. Veja exemplos de Pocket PC da HP e Dell na figura abaixo. (WINCE BRASIL, 2005).

FIGURA 2: Pocket PC HP IPaq 2210 e Dell Axim X50v - 2005
FONTE: Windows CE. Disponível em: www.wince.com.br.

4.2.1 História

No final do ano de 1996, com o lançamento do Microsoft Windows CE, iniciou a comercialização dos *handhelds*. As primeiras versões dos *handhelds* eram semelhantes a um *notebook*, contemplando inclusive teclado, mas de tamanho bem reduzido e com tela sensível ao toque. Veja o exemplo destes dispositivos na figura 3. Eles foram desenvolvidos para concorrerem com os recém lançados Palm Pilot. Mas não foram bem sucedidos, pois eram mais caros que os *Palm* e tinham um desempenho inferior além de serem difíceis de manusear. Os *handhelds* foram muito utilizados na época por suportarem que novos aplicativos fossem criados de forma simples, sendo que atualmente muitas empresas utilizam o *handheld* para fazerem seus pedidos de vendas. (*ibidem*).

No início do ano 2000, surgiu o Windows CE 3.0 e junto com ele a arquitetura Pocket PC. O Pocket PC é um s*hell,* que é uma camada de apresentação e interface entre o sistema operacional Windows CE e o usuário. Pocket PC também é o termo como ficaram conhecidos os equipamentos que utilizam o conjunto Windows CE mais o shell Pocket PC. (*ibidem*).

FIGURA 3: Handheld HP 320LX e HP Jornada 720 - 2005
FONTE: Windows CE. Disponível em: www.wince.com.br.

4.2.2 Recursos

O Pocket PC é muito parecido fisicamente com um Palm, normalmente não contém teclado, tendo como dispositivo de entrada a caneta, que é usada na tela sensível ao toque. Eles são fabricados por várias empresas, mas o que determina este equipamento ser conhecido como Pocket PC ou não, é a utilização do Windows CE com *Shell* Pocket PC. (PALM LAND, 2005).

O Pocket PC é um equipamento com tela colorida, que varia de 256 cores a 65.000 cores, em resoluções de 320 x 240 ou superior. Os equipamentos mais sofisticados contêm telas VGA com resolução de 640 x 480. Oferecem recurso multimídia, com reprodução de som, vídeos, gravação de voz e alguns modelos possuem câmera digital. (PALM BRASIL, 2005).

Para armazenamentos contam com memória interna que varia entre 32MB até 256MB. Esta memória interna pode facilmente ser ampliada através de um cartão MMC ou SD. (*ibidem*).

Além de possuírem várias formas de comunicação, podendo ser via cabo USB/Serial, *Bluetooth*, Infravermelho, *Wireless*. (*ibidem*).

Possibilitam alguma forma de expansão através de cartão MMC ou similar. Este cartão possibilita o aumento de memória para armazenamento ou a inclusão de um novo periférico, como por exemplo, um equipamento que não possua *Wireless*, poderá contar com este recurso com a utilização de cartão MMC *Wireless.* (*ibidem*).

O Pocket PC contém bateria recarregável, que tem uma duração que varia entre fabricante e modelo, mas com média de 8 horas. Na maioria dos Pocket PC a bateria pode ser substituída de maneira fácil, o que possibilita ter uma bateria reserva. (WINCE BRASIL, 2005).

Nem todos os modelos contam com os recursos citados, pois isto pode variar de fabricante para fabricante. (*ibidem*).

Existem também versões de Pocket PC com telefone celular embutido, que são conhecidos como Smartphone PC. (*ibidem*).

5 SISTEMAS OPERACIONAIS DOS DISPOSITIVOS MÓVEIS

Como ocorre nos computadores, os *handheld* necessitam de um sistema operacional para realizarem uma interface entre o hardware e software. O sistema operacional é responsável por controlar todas as funções do *handheld*, desde controle sobre a bateria, interface, sincronização com o computador, formas de comunicação, até reconhecimento de escrita. (PALM LAND, 2005)

5.1 PALM OS

A primeira idéia que se teve de um computador de mão foi para substituição de um bloco de notas de papel, por um bloco de notas eletrônico, sendo o que você escreve é armazenado de forma digital neste meio eletrônico. Essa foi a idéia aplicada no filme "Star Trek" ("Jornada nas Estrelas") que na espaçonave Enterprise, nenhum membro da tripulação utilizava blocos de papel, lápis ou caneta. Os tripulantes tinham de usar o *handheld* Tricoder e comunicadores para coletarem dados e transmitir mensagens. (TROIS, 2003).

5.1.1 História

A história do Palm se mistura com a vida de seu idealizador Jeff Hawkins, o qual se formou em engenharia elétrica na Universidade Cornell em 1979, no entanto o que chamou a atenção de Hawkins durante seus anos de faculdade foi pelo funcionamento do cérebro humano. Depois de formado, e de ter trabalhado alguns anos, ele voltou à faculdade e fez PHD sobre a teoria de memória associativa. Durante seus estudos ele criou o sistema de reconhecimento da escrita que serviu

de base para todo sistema de reconhecimento usado hoje nos p*almtops*. Hawkins voltou a trabalhar, pois, tinha que sustentar sua família, mas queria “ficar rico”, de forma que pudesse se dedicar unicamente ao estudo do cérebro humano. (ALEXANDRONI, 2001).

Alguns dos passos que Hawkins deu até a criação do Palm, ainda ocorrerem dados enquanto ele estava na GridPad, onde desenvolveu, junto com sua equipe, o primeiro *handheld* do mundo. Que tinha a dimensão de um caderno, tela monocromática e custava cerca de U$2350 sem software. O sistema de entrada de dados se dava pela tela sensível ao toque com reconhecimento da escrita. O produto foi desenvolvido voltado para aplicações de coleta de dados em escolas, feiras, ações policiais, entre outros. Hawkins percebeu que a idéia de dispositivos móveis tinha um grande potencial, e passou a dedicar-se a desenvolver um produto que fosse acessível para os consumidores comuns, e não somente para o mundo corporativo, sendo vendido em lojas de informática para concorrer com as agendas eletrônicas da Cassio e Sharp. O projeto foi chamado de Zoomer, que foi apresentado sem sucesso para os diretores da GridPad. (PALM BRASIL, 2005).

Foi então que em 2 de janeiro de 1992 após sair da GridPad, Hawkins criou a Palm Computing. No início a Palm Computing era uma empresa somente desenvolvedora de software, estava voltada para a produção de soluções para dispositivos portáteis, para empresas que fabricassem *handheld*s. O problema foi que nenhuma empresa se dispôs a investir em *palmtops*, então Hawkins juntamente com outros parceiros dentre eles a Cassio, resolveram lançar seu próprio produto, chamado de Zoomer. O projeto foi um fracasso, os motivos teriam sido de que o sistema era lento, o reconhecimento da escrita era considerado ruim, e pelo fato do

concorrente Newton da Apple ter sido um fracasso, isso afetou o sucesso do Zoomer. (PALM LAND, 2005).

Com o fracasso do Zoomer, Hawkins começou a desenvolver o Zoomer 2, sendo aperfeiçoado o sistema de escrita, agenda de contatos e compromissos, foi aprimorado o sistema de sincronismo com o computador, mais tarde chamado de HotSync. Mas Hawkins novamente teve dificuldades, a falta de um hardware para seu produto, então, sem outra alternativa, resolveu desenvolver seu próprio hardware, com isso, montou uma estrutura para possibilitar o desenvolvimento do hardware e melhorou o projeto do Zoomer 2. Para tanto Hawkins fez uma modificação importante em seu novo protótipo, enquanto os concorrentes procuravam fazer um sistema que reconhecesse a escrita humana, Hawkins, desenvolveu o sistema *graffiti*, sendo que o usuário deveria escrever de forma que o computador entendesse. Isto se tornou uma revolução, na área do reconhecimento da escrita. A idéia de Hawkins foi: “Se todos podem aprender a usar o teclado, porque não podem aprender a utilizar meu sistema de escrita?”. (PALM BRASIL, 2005).

Então em 1996, com uma parceria com a US Robotics, foi lançado o Pilot, o primeiro *palmtop* que fez sucesso no mercado desde o fracasso do Zoomer e do Newton da Apple. (*ibidem*).

No entanto, depois de algum tempo a US Robotics, foi comprada pela 3Com, e mais algum tempo depois a Palm foi separada da 3Com, por medidas estratégicas da empresa. (*ibidem*).

5.1.2 Recursos

O PalmOS é o sistema operacional para dispositivos móveis desenvolvido pela PalmSource. O PalmOS é um sistema leve, destinado para plataformas de baixo custo com alto desempenho, tem uma interface muito amigável, de fácil aprendizado e fácil acesso. Basicamente tem um menu com várias categorias, dentre elas ALL, MAIN, GAMES, SYSTEM, UTILITIES e UNFILED, conforme mostra a figura 4. Elas podem ser customizadas, para melhor organizar a distribuição dos aplicativos, de forma a melhor atender o usuário. (GRUPO PALM BR, 2005)

FIGURA 4: Interface do PalmOS – 2005
FONTE: PalmLand. Disponível em: http://www.palmland.com.br.

Existem alguns programas que são inerentes ao sistema Operacional do Palm, que são Address Book (Endereços), Date Book (Agenda), Memo Pad (Bloco de notas), Todo Lista(Coisas a fazer), Note Pad (notas), Calç (calculadora). (PALM LAND, 2005).

5.2 WINDOWS CE

O Windows CE é um sistema operacional da Microsoft, desenvolvido especialmente para ser utilizado em dispositivos móveis como *handheld* e Celulares. Este sistema operacional é robusto desde sua primeira versão, sendo que além de equipar os *handheld* com recursos de agenda, lista de contatos, anotações, tela sensível ao toque, possibilidade de usar editor de texto através do Pocket Word, planilhas através do Pocket Excel, inclusive acessar internet através do Pocket Internet Explorer. Também possibilitou a criação de novos aplicativos específicos. A criação de novos aplicativos deu outras utilidades ao *handheld*, como por exemplo: montar pedidos de vendas, recolherem informações de pacientes, entre outros e depois sincronizar tudo com computador, ou seja, todas as informações que estão armazenadas no *handheld* são transmitidas para o computador. Veja na figura abaixa duas telas do Windows CE. (WINCE BRASIL, 2005)

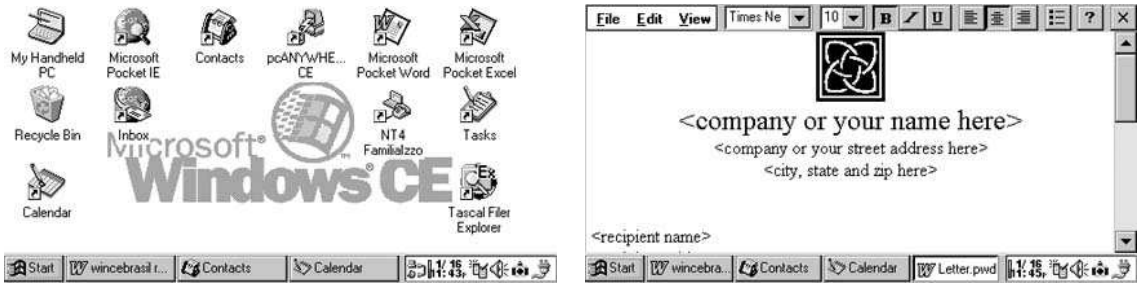


FIGURA 5: Desktop do Windows CE 1.0 e Pocket Word 1.0 - 2005
FONTE: Windows CE. Disponível em: www.wince.com.br.

5.2.1 História

A Microsoft vinha desenvolvendo 2 novos sistemas, um para máquinas de xérox e outro para TV. Para manter a compatibilidade com o Win32, a Microsoft pretendia usar uma versão de sistema operacional com o núcleo do Win NT. (*ibidem*).

Como manter a compatibilidade entre aplicações Win32 e estas plataformas eram muito difíceis, pelas limitações dos equipamentos, em Outubro de 1994 foi decidido abandonar estes projetos separados e começar um novo (com o nome de código *Pegasus*). Este projeto, que não tinha mais compatibilidade com Win32, foi sendo desenvolvido sobre um novo sistema operacional, que foi criado de forma secreta. Este novo sistema operacional deu origem ao Windows CE. (*ibidem*).

O novo sistema operacional Microsoft Windows CE, atualmente conhecido como Windows Mobile foi criado para ser usado em dispositivos de comunicações, entretenimento e computação móvel. Esta plataforma torna possível criar novas categorias de dispositivos comerciais, que podem se comunicar, compartilhar informações com PCs baseados em Windows e conectar-se à Internet. Os primeiros produtos baseados no Windows CE, começaram a ser comercializados nos EUA em novembro de 1996. O Windows CE é um sistema operacional inteiramente novo, compacto e portátil, podendo ser utilizado em PCs de mão, dispositivos de comunicação sem fio como bips digitais de informações, telefones celulares inteligentes, consoles de última geração para entretenimento e multimídia, inclusive DVD player, e dispositivos de acesso construídos para a Internet como TVs e telefones Web da Internet.. (PALM LAND, 2005).

5.2.2 Recursos

O Windows CE tem sua interface parecida com o MS Windows para computadores pessoais, contando com um menu < iniciar > onde ficam distribuídos os atalhos para acessar os programas, possibilidade de navegar entre as janelas dos programas abertos, controle de arquivos através de pastas e subpastas. Apesar de toda esta semelhança, os programas desenvolvidos MS Windows computador

pessoal não funcionam no Windows CE e vice-versa, com exceção das aplicações desenvolvidas para funcionarem com o compact framework .Net, que está disponível no Windows CE versão 4.0 ou superior. Veja logo abaixo três telas do Pocket PC. (WINCE BRASIL, 2005).

FIGURA 6: Desktop, gerenciador de arquivos e Media Player do Pocket PC - 2005
FONTE: Windows CE. Disponível em: www.wince.com.br.

O Windows CE inclui uma agenda de compromissos com calendário, agenda de contados, controlador de tarefas, bloco de notas, gerenciador de *e-mail* com possibilidade de ver *e-mail html* e receber anexo. O Pocket Word e Pocket Excel que são versões reduzidas do MS Word e MS Excel do MS Office, Pocket Internet Explorer que é versão reduzida do Internet Explorer, MSN Messenger para troca de mensagens instantâneas e o Media Player para poder ver vídeos e escutar música, sendo que é possível transformá-lo em um mp3 *player*. (WINCE BRASIL, 2005).

Ele é um sistema operacional multitarefa de 32bits, ou seja, permite executar mais de um aplicativo ao mesmo tempo. Um exemplo é ouvir-se música enquanto vê e-mails. Esta alternação entre aplicações é feito pela barra de tarefas. Mas a utilização deste recurso é o maior problema, pois exige muito do processador e faz com que esgote a carga da bateria rapidamente. (*ibidem*).

No Windows CE também é possível escrever na tela, através da caneta e ter sua letra reconhecida e convertida em texto, através do recurso que se chama Transcriber. Isto evita que o utilizador tenha que aprender uma nova forma de escrever, como ocorre no Palm. Para quem não deseja utilizar este recurso, a entrada de informações pode ser feito através de um pequeno teclado que aparece na tela e assim digitar o texto, como mostra a figura abaixo. (*ibidem*).

FIGURA 7: Transcriber e Teclado do Windows CE - 2005
FONTE: Windows CE. Disponível em: www.wince.com.br.

Para controlar a bateria existe uma API do Windows CE que fornece o tempo que a carga vai durar, podendo ser usado para avisar ao usuário que a carga da bateria está acabando ou até mesmo salvar as informações antes que o Pocket PC desligue automaticamente. Para as informações que já estão salvas não se perderem, existe uma bateria interna de *backup* que mantém os dados até que a próxima recarga na bateria seja feita. (WINCE BRASIL, 2005).

A sincronização dos dados armazenados é feito através do ActiveSync da Microsoft. O ActiveSync é um software que fica instalado no computador e tem como finalidade estabelecer a comunicação com o Pocket PC e fazer a comparação dos dados. A sincronização é feita automaticamente ao estabelecer a conecção com o computador, sendo que esta conecção pode ser feita via cabo USB/Serial ou *bluetooth*. (*ibidem*).

6 PLATAFORMAS DE DESENVOLVIMENTO

Plataformas de desenvolvimento podem ser consideradas as ferramentas ou software utilizados para o desenvolvimento de novos softwares. Estas ferramentas possibilitam ao programador toda uma estrutura, que normalmente se compõe de um editor, no qual é digitado todo o código do novo software, um compilador, que é responsável por transformar o código digitado em um outro código que é entendido pelo computador e assim possibilitar a execução do software. (FRANKLIN, 2002).

6.1 VISUAL BASIC .NET

O Visual Basic.NET é uma plataforma de desenvolvimento baseada na linguagem de programação Basic, desenvolvida pela Microsoft. Esta ferramenta foi criada para a construção de programas compatíveis com o Windows de uma forma fácil, simples e amigável. (*ibidem*).

6.1.1 História

O Visual Basic surgiu após o lançamento do Windows, que foi um marco na história, pois revolucionou a forma com que o usuário interagia com o computador. No entanto, criou-se um problema grave, pois quase não existiam programas para Windows, já que na época as ferramentas de programação disponíveis para interface gráfica do Windows eram precárias. Para se ter uma idéia da dificuldade que era desenvolver um software para Windows, somente para que se pudesse desenhar uma pequena janela, muito simples, gastava-se dezenas, talvez centenas de linhas de código, o que tornou muito difícil para os programadores, fazendo com que poucas empresas investissem no Windows. Para solucionar este problema a

Microsoft teve que criar uma forma de facilitar o trabalho do programador e para isto desenvolveu uma nova linguagem. (CRAIG, 1994).

Durante seu desenvolvimento esta linguagem era conhecida como Thunder e logo após o seu lançamento em maio de 1991 recebeu o nome de Visual Basic 1.0. (WAKEFIELD,2002)

A Linguagem Visual Basic 1.0 foi idealizada para ser uma linguagem gráfica, no entanto como costuma acontecer nas versões 1.0 de vários produtos, para manter o prazo de lançamento do produto, fez-se necessário abrir mão de vários recursos idealizados para esta versão. (LOMAX,2002).

Com o surgimento da ferramenta Visual Basic, mudou completamente a forma de desenvolvimento de softwares para Windows, pois na época eram poucas as pessoas habilitadas, dado o alto grau de dificuldade de se programar de forma visual utilizando as linguagens existentes. (*ibidem*).

Com o surgimento do Visual Basic possibilitou que muitos programadores migrassem para esta nova ferramenta, formando um ciclo de desenvolvimento e expansão do Windows. (*ibidem*).

Por causa da dificuldade de acesso a dados através do Visual Basic 1.0 em novembro de 1992, a Microsoft lançou o Visual Basic 2.0, contendo um suporte melhor a banco de dados através do ODBC, acrescentou também uma janela de propriedades, sintaxe de código em várias cores e completo suporte a múltiplos documentos (MDI – Multiple Document Interface). (*ibidem*).

A versão 3.0 do Visual Basic foi lançada após seis meses do lançamento da versão 2.0, para atender as demandas de uma ferramenta de acesso a banco de dados, pois a versão 2.0 não atendeu todas as necessidades dos programadores. Esta nova versão veio com um mecanismo de banco de dados do Access 1.1 com

um sofisticado conjunto de controles de acesso a dados, permitindo pela primeira vez, que se fosse aplicada a funcionalidade de um ambiente cliente/servidor numa linguagem intuitiva e visual. Outra funcionalidade adicionada à versão 3.0 foi o Crystal Reports, permitindo que os dados recuperados possam ser apresentados sobre uma grande variedade de formatos personalizáveis de relatórios. (FRANKLIN, 2002).

Em setembro 1995 foi lançado o Visual Basic 4.0, para atender as novas demandas da computação de 32 bits, com os respectivos lançamentos do Windows 95 e NT. Umas das principais funcionalidades adicionadas foi o controle de dados remoto e um software integrado para o gerenciamento de configuração de versões, chamado Microsoft Visual SourceSafe. (WAKEFIELD,2002).

Com o surgimento da versão 4.0, que é orientado a eventos, se ganha uma nova possibilidade de utilizar como uma linguagem orientada a objetos. Através do Jet Engine, que acompanha esta nova versão, praticamente foram resolvidos os problemas com bancos que o Visual Basic possuía. (*ibidem*).

Em março de 1997 e junho de 1998 surgiram respectivamente as versões 5.0 e 6.0 do Visual Basic. Nestas versões alguns destaques são o grande aumento de performance do compilador nativo de código, chegando a uma melhora de até 2.000%. (LOMAX,2002).

O Visual Basic .NET surgiu após o lançamento do Microsoft .NET Framework, que surgiu a partir de uma evolução de idéias, que teve como resultado uma nova tecnologia, considerado também um novo conceito no desenvolvimento de aplicativos. (FRANKLIN, 2002).

6.1.2 Tecnologia .Net

A tecnologia .NET baseia-se numa estrutura de camadas de serviços responsáveis pelo controle de aplicações, que executam sobre sua estrutura. Ela possui uma grande biblioteca de código, que substitui as atuais API do Windows. O .NET framework tem sua idéia principal, muito parecida com a da plataforma Java, que utiliza uma máquina virtual. No entanto a idéia do .NET framework se mostra mais abrangente e ambiciosa, pois dá abertura para que muitas linguagens possam ser compiladas e que rodem sobre a sua estrutura, ou seja, ele traz uma independência na linguagem de programação. (FRANKLIN, 2002).

6.1.3 Principais Recursos do Framework

- **Assemblies:** No ambiente Windows sempre houve problemas quanto a DLLs que são utilizadas na maioria das aplicações. Estes problemas incomodavam tanto, que chegaram a ser chamadas de “o inferno das DLLs”. Um caso muito comum para exemplificar-se este problema era o de uma aplicação que tinha uma DLL substituída por uma outra DLL de mesmo nome, mas de versão ou desenvolvedor diferente. Isto trazia como conseqüência, na maioria dos casos, problemas no funcionamento do software que teve a DLL substituída, com isso, tinham que se instalar novamente o software, para que os programas voltassem a funcionar, e às vezes causando problemas. (WAKEFIELD,2002).

  Mas o. NET framework foi um pouco, além disso, com a criação dos *assemblies* que de certa forma substituem os DLLs ou EXE. Um assemblie é o conjunto de arquivos físicos, que na maioria contêm dados, como pedaços

de códigos, classes criadas, mas que também podem conter outros tipos de arquivos, como fotos, por exemplo. (*ibidem*).

> No .NET Framework, o código gerenciado é implantado na forma de assemblies. Você pode considerar os assemblies como análogos às DLLs ou EXEs do VB, embora sejam muito mais do que simples DLLs ou EXEs, que são basicamente arquivos binários. (FRANKLIN, 2002, p. 5)

Os *assemblies* podem ser de dois tipos: Privados e Compartilhados. Os *assemblies* privados, são restritos ao programa que os criou, para serem usados por ele, com isso minimizam consideravelmente o problema com DLLs pois desta forma as aplicações tem seus componentes protegidos. Já os *assemblies* compartilhados podem ser acessados por todas as aplicações da máquina, ou seja, quando um programa for executado, se tentará encontrar um *assemblie* privado, mas caso não o encontre, então será procurado por um *assemblie* compartilhado para aquela aplicação. (*ibidem*).

- **Metadados:** O .NET framework foi criado sobre a perspectiva de que o código deve ser auto descritivo, de forma que outras ferramentas de desenvolvimento através dos metadados poderão saber a forma de interagir com o programa. (WAKEFIELD,2002).

Os metadados também são importantes para o acesso remoto a dados, onde são definidas as formas de acesso aos dados, isto é, se você vai ter acesso direto aos dados, ou se os dados terão de ser criptografados antes de mandar pela rede e descriptografados na saída. (*ibidem*).

Sendo assim os metadados podem ser usados para localizar e carregar classes, gerar código nativo e fornecer segurança, pois são neles que se encontram as informações para os programas, tais como no caso da rede, é ele quem da às informações sobre como os dados devem trafegar na rede, desta forma ele guarda informações básicas dos programas. (*ibidem*).

- **Common Language Runtime (CLR):** O CLR é basicamente o ambiente de execução das aplicações .NET. Ele é responsável pela interação com o sistema operacional. (LOMAX,2002).

  Também cuidará do gerenciamento de memória através do *garbage collector*, que percorre a memória limpando as partes correspondentes a dados que não estão mais em uso, melhorando assim a performance e evitando erros no sistema. (*ibidem*).

  Quando um código é compilado para CLR ele é chamado de código gerenciado, isto é, o código contendo os metadados, com as informações de tipos, membros, referências no código, entre outros. (*ibidem*).

  O CLR é responsável pelas seguintes tarefas dentro do Framework:

    - Sistema de tipos comuns (Comum Type System)
    - Coleta de lixo (Garbage Collector)
    - Compilação Just-In-Time (explicado a seguir)
    - Controle de versões (explicado a seguir)
    - Segurança

- **Compilação Just-In-Time (JIT):** Em alguns ambientes de desenvolvimento o código é compilado somente na hora da implantação, mas com o surgimento do *byte code* do JAVA, que retarda a compilação até o momento da execução, possibilitou a execução do código em várias plataformas diferentes sem a necessidade de vários pacotes de instalação diferentes. (FRANKLIN, 2002).

  Quando um software é feito através do Visual Basic .NET e é compilado para a linguagem IL, este código é independente de plataforma,

podendo assim funcionar em qualquer plataforma que tenha o CLR instalado no computador. (*ibidem*).

Então a partir do momento que o código é executado inicia o processo do compilador JIT. O CLR verifica se este código já foi compilado e armazenado em cache e então será verificado se o código é *type-safe*, ou seja, implementada a segurança de tipos através do CLR e o compilador JIT. Portanto a função básica do JIT é converter as instruções de IL, para instruções específicas da linguagem de máquina do computador em que o código está sendo executado. (*ibidem*).

Este compilador foi projetado para máquina de baixo desempenho, com poucos recursos de CPU, como os *handhelds*, por exemplo. Também mantêm um registro dos códigos, para eliminar os códigos menos utilizados, e voltar a compilar, caso sejam necessários, de forma a economizar memória de execução do programa. (*ibidem*).

- **Controle de Versões:** No .NET framework o controle de versões é usado para indicar a que versões de assemblies a aplicação terá acesso. Esse controle é feito de acordo com uma política de versões. Cada assemblie tem um número de 4 dígitos, segundo o quadro abaixo:

QUADRO 13: CONTROLE DE VERSÃO NO .NET - 2002

Primário	Secundário	Construção	Revisão
2	0	2	11

FONTE: FRANKLIN, p. 9

> O .NET Framework fornece uma política padrão de versões que pode ser anulada. Essa política especifica que o assembly a ser carregado deve ter números de versão primários e secundários iguais e selecionara os maiores números de construção e revisão. Tal política padrão assume o seguinte:

> - Números de versão primários diferentes não são compatíveis.
> - Números de versão secundários diferentes geralmente não são compatíveis.
> - Números de construção (builds) diferentes devem ser compatíveis.
> - Números de versões diferentes devem ser compatíveis.
>
> (FRANKLIN, 2002, p. 5)

O controle de versões pode ser garantido pelo uso da política padrão ou um novo conceito de padrão pode ser criado. Desta forma se pode ter um controle personalizado, sobre as versões de forma a se adaptar as necessidades. (*ibidem*).

- **Namespaces:** Os *namespaces* são utilizados para disponibilizar componentes, como classes, estrutura, interface e entre outros, os quais são utilizados e organizados, porque durante o uso de COMs existia apenas dois níveis, o nome do componente e a classe dele, no entanto com o .NET se pode ter muitas combinações. (WAKEFIELD,2002).

  As aplicações COM usavam o GUID (*Global Unic Identifier*), usado como identificador exclusivo para os componentes, de forma que os nomes são limitados à associação com o nome da biblioteca de uma classe. (*ibidem*).

  Já no .NET os GUIDS são substituídos por namespaces, dando uma organização mais lógica diminuindo as limitações de nome. No .NET os componentes são vinculados a um namespace. (*ibidem*).

- **Biblioteca de classes do .NET Framework:** O .NET framework disponibiliza uma grande variedade de classes e elementos prontos para o uso com linguagens compatíveis com CLR. Estes elementos podem ser sistema de janelas, biblioteca de entrada e saída de dados, acesso a funções de I/O, gerenciamento de memória, acesso a dados e de segurança do .NET. Com

elas é possível se obter muitos recursos prontos que antes tinham que ser desenvolvidos a partir do zero. (LOMAX,2002).

A biblioteca de classes fica inclusa dentro do *namespaces*, e dentro do *namespace* System ficam todas as classes que representam todos os tipos de dados básicos utilizados para construção de aplicações. (*ibidem*)

6.1.4 Recursos do Visual Basic .NET

O Visual Basic .NET é uma atualização, quase completa do Visual Basic, cuja principal alteração, assim como ocorreram nas demais linguagens que incorporaram o conceito do .NET, foi a troca da utilização da API do Windows pela .NET Framework. (FRANKLIN, 2002).

Durante muitos anos a linguagem Visual Basic passou por muitas mudanças, por nunca ter passado por um órgão definidor de padrões, facilitando o trabalho da Microsoft no desenvolvimento da ferramenta, mas por outro lado foi péssimo para os programadores que tem que aprender tudo de novo praticamente a cada versão. No entanto, isso parece tender a mudar agora com o surgimento do .NET, mas não existem garantias. (*ibidem*).

Os objetivos do projeto Visual Basic .NET, seguem uma linha de programas funcionais com Web Services, ou seja, que sejam fáceis de criar aplicações compatíveis com serviços da internet. (WAKEFIELD,2002).

Dentre os objetivos definidos para a linguagem, pode-se destacar uma sintaxe simples e direta através do uso dos recursos básicos do Framework .NET, facilidade nas atualizações de aplicações do Visual Basic 6, mantimento da linguagem, na medida do possível, parecido com as versões anteriores. (*ibidem*).

O Visual Basic, desde seu início, foi conhecido por ser acessível, o que atraiu muitos programadores iniciantes para desenvolverem programas em VB. (*ibidem*).

Uma das vantagens do Visual Basic, é que ele facilita o programador menos experiente pelo fato de permitir a declaração de variáveis, sem declarar o seu tipo, e isto de certa forma ajuda muito, mas também pode causar erros difíceis de serem detectados. (LOMAX,2002).

Outra característica importante do Visual Basic é o fato dele ser uma linguagem simples, ele evita o uso de caracteres e operadores especiais, isto é feito através de palavras e comandos. Com isso, torna o código mais longo, sendo que numa linguagem C, por exemplo, teria de se escrever menos. Mas facilita na documentação, pois é mais fácil de entender o código, não necessitando de muitos comentários, como ocorrem em outras linguagens. (*ibidem*).

Desde o princípio da Orientação a Objetos, esta foi uma meta perseguida para a linguagem Visual Basic, que teve implementado na sua versão 4, na versão 5 obteve vários recursos, mas tornou-se efetivamente de suporte integral na versão .NET. (FRANKLIN, 2002).

Na versão .NET também é desenvolvida a idéia de herança, ou seja, o programador pode em sua aplicação Visual Basic .NET além de herdar classes de outras aplicações Visual Basic .NET, pode também herdar classes de outras linguagens compatíveis com .NET. (*ibidem*).

Foi adicionada a esta nova versão a manipulação estruturada de execuções que deve facilitar o tratamento de erros. (*ibidem*).

O Visual Basic .NET como todas as ferramentas que utilizam a tecnologia do .NET Framework, vem com uma infra-estrutura de segurança para as aplicações,

sendo possível, de forma fácil, proteger dados e processos de acessos inadequados. (LOMAX,2002).

6.1.5 Compact Framework

O .NET Compact Framework (.NET CF) é um parte do .NET framework e tem como objetivo prover um ambiente para desenvolver aplicações de forma fácil para dispositivos que utilizam Windows CE. O benefício é que os programadores tenham a mesma facilidade no desenvolvimento de aplicações para dispositivos móveis utilizando a mesma IDE e qualquer uma das linguagens que dispõe da tecnologia. NET. Entretanto não são todas as classes e métodos que se encontram no .NET framework que são suportadas dentro do .NET compact framework. A seguir é possível verificar a imagem 8, que mostra de forma mais clara quais as classes que estão disponíveis somente no .NET compact framework. (FRANKLIN, 2002).

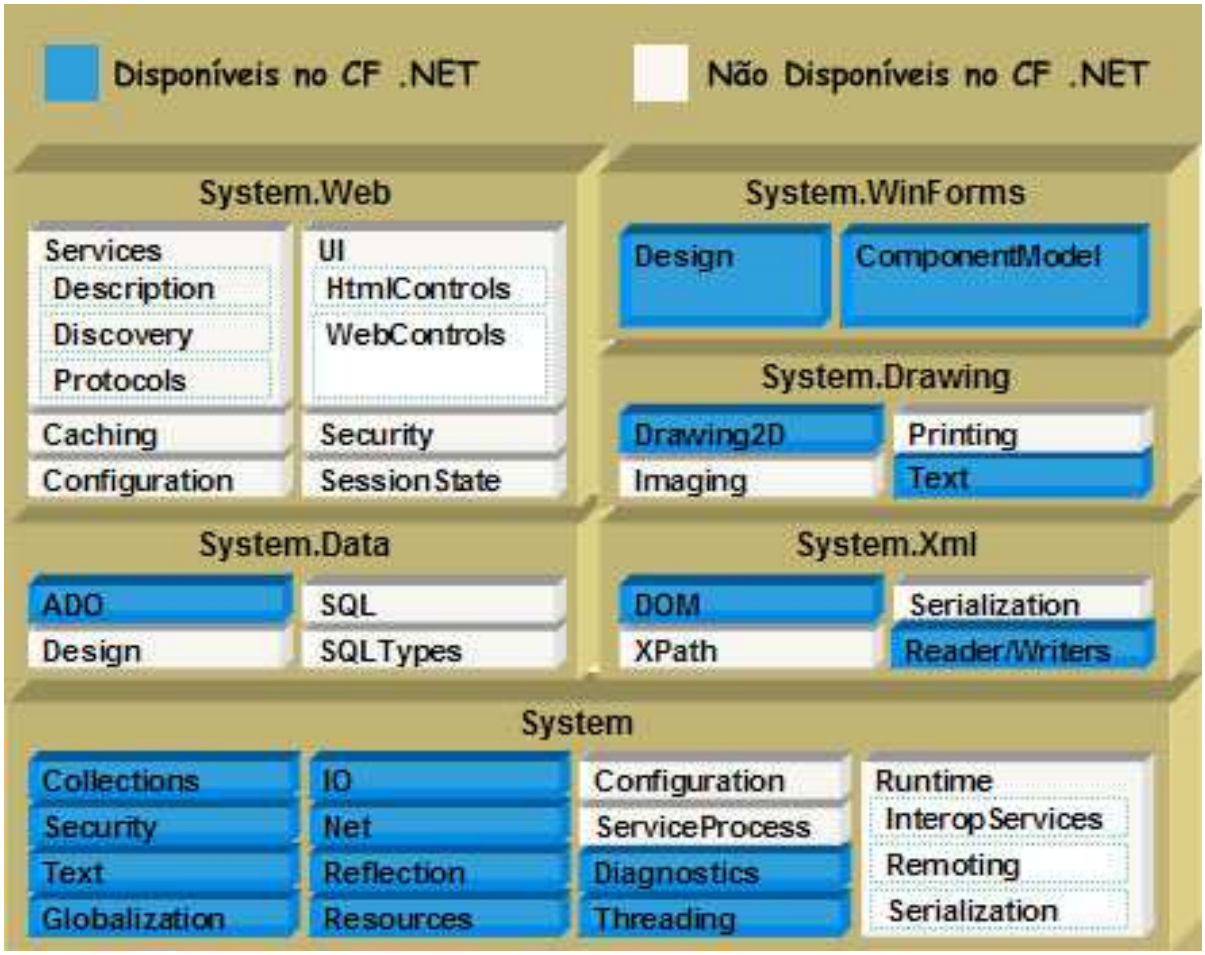


FIGURA 8: Classes do Compact Framework .NET - 2005
FONTE: Disponível em: http://www.activevb.de/rubriken/kolumne/kol_3/compactframework.html

6.2 POCKETSTUDIO

O PocketStudio foi desenvolvido especialmente para fazer com que os programadores Delphi programassem também para PalmOS, sem muitas dificuldades de adaptação. (ALEXANDRONI, 2001).

6.2.1 História do PocketStudio

PocketStudio é a primeira ferramenta a utilizar a Linguagem Pascal como base. Sua interface é visual, sendo muito parecida com o Delphi, o que fez se tornar uma ferramenta muito esperada por estes programadores. (*ibidem*).

Esta ferramenta começou e ser desenvolvida em 1999 pela empresa Pocket-Technologies, que está sediada no estado de Minnesota nos Estados Unidos. (*ibidem*).

> Em meados de 2000 já contavam com um compilador rápido e poderoso, além de gerar executáveis nativos para o PalmOS. Após a criação do compilador, foi iniciado um período de testes que durou em torno de um ano. Em agosto de 2001 foi lançado oficialmente o PocketStudio. (ALEXANDRONI, p. 23)

6.2.2 Características do PocketStudio

O PocketStudio possibilita a criação de executáveis nativos PalmOS, com acesso a API do sistema operacional. Durante o início de um novo projeto, esta ferramenta possibilidade indicar para qual versão do PalmOS o software vai ser desenvolvido, o que pode trazer várias vantagens, já que um software vai funcionar em várias versões de sistema operacional sem a necessidade de nenhuma alteração. (ALEXANDRONI, 2001).

Esta ferramenta fornece também um *debug* diretamente no emulador, o que facilita o descobrimento de erros no software. A utilidade do emulador também possibilita criar um software sem mesmo possuir um *handheld* Palm, já que todos os testes podem ser feitos diretamente no emulador. (*ibidem*).

Possui acesso a banco de dados Oracle Lite e Palm DataBase, que é um banco de dados nativo do PalmOS, além de suporte a tecnologias auxiliares, como *conduits* que é responsável pela comunicação com o computador, *bluetooth* e *wireless* para comunicação sem fio com outro equipamentos. (*ibidem*).

6.3 JAVA

A Linguagem Java é uma linguagem de programação que foi orientada a objetos desde o seu surgimento. Na Linguagem Java, tudo é uma classe, com propriedades e métodos. No momento de seu desenvolvimento, seu criador James Gosling, sabendo que a Linguagem C e C++ eram de alto conhecimento entre os programadores da Sun, resolveu fazer esta nova linguagem com uma sintaxe básica parecida com a Linguagem C++. (RITCHEY, 1997).

6.3.1 História

Em 1991 a Sun Microsystems, empresa criadora da Linguagem Java, apoiou alguns desenvolvedores para que fizesse algo inovador, da maneira que eles achassem melhor. Estes desenvolvedores resolveram criar algo para integrar digitalmente produtos distintos, como TVs, CD players, computadores e outros aparelhos. Durante este desenvolvimento, várias tecnologias foram testadas ou até mesmo inventadas. Uma deles foi a Linguagem C++, que é uma boa linguagem de programação, mas não oferecia portabilidade e tamanho pequeno o suficiente para caber em dispositivos pequenos. Para resolver estes problemas, surgiu o Oak, que é uma linguagem criada pelo James Gosling e no início foi utilizado somente internamente pela Sun Microsystems. (*ibidem*).

Após perceber que a nova linguagem teve um potencial muito grande, ela teve seu nome trocado para Java. (*ibidem*).

Um ano após a sua criação, foi lançado um pequeno dispositivo com uma pequena tela. Este dispositivo foi chamado de Star-7 e era um *handheld.* Para deixar o uso deste equipamento mais interessante, foi criado um personagem interativo,

chamado Duke, que acabou se tornando a mascote oficial da Linguagem Java. Tanto o criador do Java, como o primeiro dispositivo criado pode ser observado na figura abaixo. (*ibidem*).

FIGURA 9: James Gosling e o handheld Star-7 - 2005
FONTE: Disponível em: www.sun.com.

No início do ano de 1993, surgiu uma boa oportunidade para a Linguagem Java, pois poderia vir a equipar equipamentos conversores de TV a cabo. Apesar desta oportunidade, a empresa que precisava destes serviços acabou por utilizar outra tecnologia. Outras oportunidades surgiram para a utilização da Linguagem Java, mas nenhuma foi dada continuidade. (*ibidem*).

Com estas dificuldades, vários membros abandonaram o projeto desta linguagem, mas nesta mesma época, surgiu um programa chamado Mosaic que utilizava o protocolo HTTP e possibilitava a visualização de documentos HTML através da internet. Com o surgimento deste programa, os membros restantes resolveram criar um programa parecido, mas com uma interface interativa, segura, robusta e independente de arquitetura, ou seja, dando a possibilidade de ser

executado em qualquer sistema operacional ou até mesmo em outros equipamentos digitais. (*ibidem*).

Desta forma, durante o ano de 1994, foi criado um *browser* chamando de HotJava, que possuía uma forma dos usuários interagirem com as páginas HTML. Entre estas interatividades, havia coisas interessantes, como objetos em 3D animados que se movimentavam conforme o mouse era movimentado. (*ibidem*).

> Em seguida, Arthur Van Hoof, que tinha passado a integrar a equipe da Sun um ano antes, implementou o compilador Java usando a própria Linguagem Java, quanto que o compilador original desenvolvido por Gosling tinha sido implementado em C. Esse fato mostrou claramente que o Java era uma linguagem completa, e não meramente um brinquedo extremamente simplificado. (RITCHEY, 1997, p. 27)

No ano de 1995, a Sun Microsystems teve uma atitude importante disponibilizando a Linguagem Java gratuitamente a fim de ganhar a atenção dos desenvolvedores de todo o mundo. Esta atitude foi um sucesso, pois em pouco tempo os membros da equipe já passavam dias e noites respondendo e-mails, consertando erros e preparando novas versões. (*ibidem*).

6.3.2 Funcionamento do Java

Como foi criada para desenvolvimento de pequenos aparelhos eletrônicos, a Linguagem Java mostrou-se ideal para ser utilizada na internet. O que torna ele tão atraente para este tipo de aplicação é que os programas desenvolvidos podem ser executados virtualmente em qualquer plataforma, ou seja, ele é um idioma comum. (KNUDSEN, NIEMEYER, 2000).

Um software desenvolvido na Linguagem Java, após ser compilado, tem seu código fonte transformado em byte code, também conhecido como *j-code*. Este *byte code* é interpretado depois pela JVM (Java Virtual Machine), que nada mais é que

uma máquina virtual que entende todo o conteúdo do *byte code* e converte para o sistema operacional em que está sendo executado. O JVM pode ser via software, ou seja, é instalado na plataforma aonde deve ser executado ou via hardware através de um microchip, que permite o *byte code* ser executado muito mais rápido. (*ibidem*).

Com a existência da máquina virtual, o *byte code* é gerado apenas uma vez, pois a máquina virtual de cada plataforma vai se encarregar do restante, que é a execução na plataforma desejada. Esta portabilidade não é possível na Linguagem C, já que esta linguagem necessita que exista uma compilação própria para cada plataforma. (*ibidem*).

Atualmente existem compiladores que convertem o *byte code* em instruções nativas de máquinas, como por exemplo, o Just In Time Compiler (JIT), que torna o software desenvolvido mais rápido do que se estivesse sendo executado através da máquina virtual. Este compilador precisa de uma versão específica para plataforma em que o programa vai ser executado e a velocidade de execução ainda vai depender da quantidade de memória, pois um *byte code* compilado pelo JIT fica em torno de três vezes maior. Na figura 10 mostra a estrutura lógica de funcionamento do Java. (SCHÜTZER, MASSAGO, 1997).

Apesar da Linguagem Java inicialmente ter sido direcionada para as aplicações *applets*, que possibilitam interações em documentos HTML, esta linguagem oferece muitos outros recursos para construção de aplicações *stand-alone,* que são independentes do ambiente de internet. Como é independente da internet, esta linguagem pode ser usada em desenvolvimento de softwares completos, com acesso a banco de dados, utilizando altos recursos gráficos e entre outros. Ela também é utilizada em desenvolvimento de software para equipamentos como celular, MP3 *player*, *handhelds*. (*ibidem*).

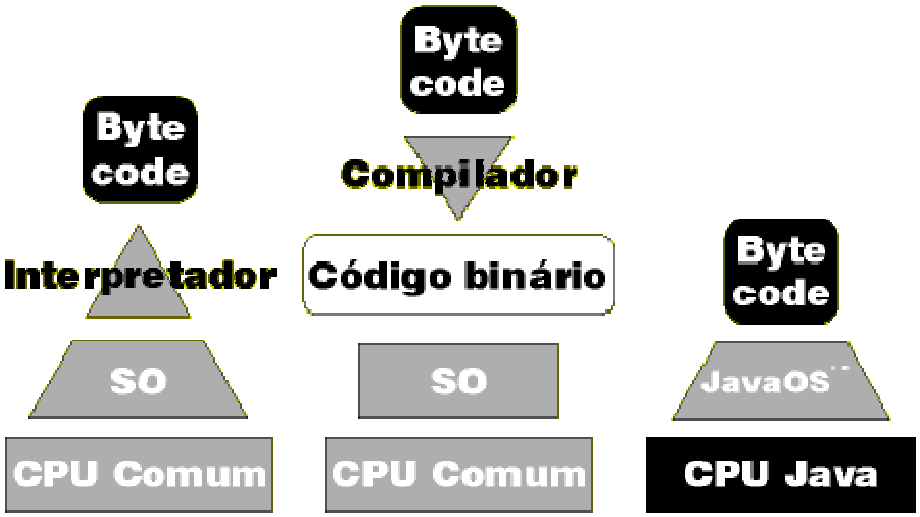


FIGURA 10: Estrutura do Java - 2005
FONTE: Disponível em: http://www2.dm.ufscar.br/%7Ewaldeck/curso/java/.

6.3.3 Características do Java

A Linguagem Java fornece diretamente em sua arquitetura muitas funcionabilidade, que tornam esta linguagem muito completa. Segue a seguir suas características:

- **Parecido com a Linguagem C++**: Como foi baseada na Linguagem C++ durante seu desenvolvimento, torna a Linguagem Java uma boa opção para os programadores da Linguagem C++ que necessitam de portabilidade, mesmo que a filosofia da linguagem seja diferente. A Linguagem Java também retira do programador as tarefas que podem levar ao erro, como gerenciamento de memória e ponteiros. Isto significa que os programadores podem gastar menos tempo se preocupando com erros de execução do código e dedicar este tempo no desenvolvimento de funcionabilidade. A Linguagem Java também possui características herdadas de muitas outras linguagens de programação: Objective-C, Smalltalk, Eiffel, Modula-3, entre outros. Muitas das características desta linguagem não são totalmente novas.

Java é uma feliz união de tecnologias testadas por vários centros de pesquisa e desenvolvimento de software. (RITCHEY, 1997).

- **Portabilidade**: O software desenvolvido em Java pode ser executado em qualquer equipamento que possua um interpretador Java instalado. Como a Linguagem Java é interpretada, indica que cada computador que pretende utilizá-la deve ter um interpretador instalado. Várias plataformas distintas podem ser utilizadas, como qualquer versão do Windows, superior ou igual ao Windows 95, Linux, Solaris, MAC/OS entre outros. (*ibidem*).

  Segundo RITCHEY, a Linguagem Java não permite que equipamentos particulares implementem tamanhos diferentes para tipos fundamentais como *integers* ou *bytes*. (1997, p. 9).

- **Robustez**: A Linguagem Java é robusta, pois consegue resolver os problemas dos programadores que precisam desenvolver um programa e que necessita ser portátil. A Linguagem Java possui um controle automático de memória e não necessita da utilização de ponteiros, o que é o maior causador de problemas ao desenvolver um programa usando a Linguagem C++. Este controle é o *garbage collector* que varre a memória usada pelo aplicativo liberando a memória que não está mais sendo utilizada. (*ibidem*).

  Segundo RITCHEY, essas características oferecidas pelo C++ permitem que programadores criem códigos corretos do ponto de vista sintático e semântico, mas que ainda causem travamentos dos sistemas em determinadas situações. (1997, p. 10).

- **Orientação a objetos**: A Linguagem Java é orientada a objetos desde sua base, permitindo todas as características como instância de classes, herança,

polimorfismo, como exceção de herança múltipla. Por ser orientada a objetos, possibilita o reaproveitamento de código, trazendo também um dinamismo durante a programação. (*ibidem*).

- **Alto desempenho**: Mesmo a Linguagem Java sendo interpretada durante sua execução, sendo natural que se tenha perda de desempenho, ainda existem recursos para amenizar este problema. Um deles é o *multithreading* que permite utilizar o processador mesmo quando ele está sendo utilizado pelo programa. Nestes casos o processador é utilizado para melhorar o gerenciamento de memória, como liberar as variáveis alocadas que não estão mais sendo utilizadas. Outro recurso importante é poder utilizar a compilação just in time, que possibilita converter o *byte code* em programa nativo para ser executado em uma determinada plataforma. Também se deve considerar que o *byte code* é muito próximo das linguagens de máquina, o que faz com que esta perda de desempenho seja reduzida. (*ibidem*).
- **Facilidade**: Por ser derivada da Linguagem C++, o seu código tem muitas semelhanças, o que torna fácil para programadores que já estão acostumados com o C++. Além disso, a Linguagem Java retira do programador algumas responsabilidades sujeitas ao erro, como gerenciamento de memória e ponteiros. (*ibidem*).

6.3.4 SuperWaba

O SuperWaba é uma plataforma de desenvolvimento de software para PDA e *smartphones*. Esta plataforma foi criada no ano 2000 e é derivado de um projeto chamado Waba. O SuperWaba é muito parecido com o Java, tanto em sua linguagem, quanto no modo de funcionamento, que utiliza uma máquina virtual. (HAZAN, 2000).

Apesar do SuperWaba não ter nenhuma ligação com a Sun Microsystems, criador do Java, esta ferramenta foi desenvolvida para ser utilizada por programadores Java. Inclusive, usando as mesmas ferramentas de desenvolvimento, mas tendo um foco para dispositivos móveis. Sendo que o SuperWaba também disponibiliza uma máquina virtual, possibilitando que o software desenvolvido utilizando esta plataforma seja executado nos seguintes sistemas operacionais: PalmOS, Windows CE e Symbian. (*ibidem*).

7 METODOLOGIA

Para realizar a análise das linguagens de programação: SuperWaba, PocketStudio e Visual Basic .NET, que são utilizadas em desenvolvimento de softwares para dispositivos móveis e realizar uma comparação, foram utilizados os critérios conforme foram abordados no capítulo 2. Também foi avaliado o desempenho de cada linguagem através de testes realizados diretamente em dois *handhelds,* sendo um modelo Palm Zire71 utilizando o sistema operacional PalmOS e outro modelo HP IPaq1940 utilizando o sistema operacional Windows CE com o *shell* Pocket PC.

Para poder avaliar as linguagens conforme o capítulo 1 foram criados tópicos de comparação contendo todos os métodos utilizados para cada uma das linguagens.

Para avaliar o desempenho das linguagens foi desenvolvido um software que tem como objetivo realizar alguns testes, divididos em três tipos: Desempenho de VÍDEO, MATEMÁTICO, MEMÓRIA.

Para cada vez que um dos testes for realizado, independente do tipo que ele pertence, foi medido seu tempo de execução em segundos, que é a diferença entre a hora inicial e a hora final. Com a soma de tempo de cada tipo de teste, foi calculado o tempo total do teste. Para avaliar melhor cada um dos testes, os mesmos foram divididos em sub-testes, como mostra a relação a seguir:

- Desempenho de VIDEO
    1. Mudar cores na tela;

       É criado um quadrado de tamanho 160 por 160 pixels, onde dentro de um laço de repetição ele tem suas cores trocadas de forma randômica por 3.000 vezes.

2. Criar círculos coloridos na tela;

   Dentro de uma área de tamanho 160 por 160 pixels, são desenhadas circunferências de forma a preencher toda esta área da esquerda para a direita, de cima para baixo, onde dentro de um laço de repetição ele tem suas cores trocadas de forma randômica por 3.000 vezes.

- Desempenho MATEMÁTICO
  1. Função para cálculo de fatorial;

     É calculado o fatorial do número 700.000.

  2. Função para cálculo do número PI;

     É executada a função para cálculo do número PI por 700.000 vezes, onde a cada execução desta função, maior é sua precisão.

  3. Função para cálculo de número primo;

     É calculado se o número 700.000 é primo ou não, através da divisão pelos números consecutivos de 1 até 700.000.

- Desempenho MEMÓRIA
  1. Salvar 2.000 registros num arquivo de dados;
  2. Ler 2.000 registros de um arquivo de dados;

Este software foi implementado de forma igual em cada linguagem para cada sistema operacional, de forma compatível, ou seja, o PocketStudio para o PalmOS, o Visual Basic .NET para Windows CE e SuperWaba para ambos os sistemas operacionais. Pelo fato da linguagem de programação SuperWaba funcionar através de uma máquina virtual, que está disponível para os dois sistemas operacionais abordados, é possível utilizar o SuperWaba como base de comparação, pois com base no desempenho nos dois dispositivos móveis, será

possível definir a diferença de desempenho entre uma linguagem nativa e uma linguagem interpretada.

A avaliação do desempenho de cada linguagem foi realizada da seguinte maneira:

- Primeiramente foi executado o programa de testes desenvolvido com a linguagem SuperWaba nos dois sistemas operacionais. O tempo final obtido em cada sistema operacional definiu qual dos dois equipamentos está tendo um melhor desempenho.
- Em segundo lugar foi executado o programa de testes desenvolvido com a linguagem Visual Basic .NET para o sistema operacional Windows CE, obtendo assim o tempo médio final para esta linguagem. Com base no tempo final obtido com a linguagem SuperWaba para o sistema operacional Windows CE foi possível analisar se a linguagem nativa obteve um desempenho consideravelmente superior a linguagem interpretada.
- Por último foi executado o programa de teste desenvolvido com o PocketStudio para o sistema operacional PalmOS, obtendo assim o tempo final para esta linguagem. Com base no tempo final obtido com a linguagem SuperWaba para o sistema operacional PalmOS foi possível analisar se a linguagem nativa obteve um desempenho consideravelmente superior à linguagem interpretada.

Com as comparações de desempenho juntamente com as comparações das linguagens de desenvolvimento foi possível ajudar um programador quanto a escolha de qual sistema operacional o *handheld* deverá conter e qual linguagem de programação é mais vantajosa para o desenvolvimento de um software.

8 ANÁLISE COMPARATIVA

Neste capítulo é abordada toda a parte da análise comparativa e do desenvolvimento do protótipo, para avaliar o desempenho das linguagens e auxiliar na escolha de uma ferramenta de desenvolvimento.

8.1 FLUXOGRAMA

O Fluxograma, que está disponível no apêndice 1, demonstra como vai funcionar o protótipo. Está dividido em 4 partes. A primeira parte, chamada de Protótipo Desempenho, mostra a ordem lógica de funcionamento, ou seja, qual a ordem de testes que é aplicada. A segunda, parte, com o nome de Teste de Vídeo, é um sub-processo do Protótipo Desempenho, que mostra a ordem em que serão executados os testes de vídeo. Como a segunda parte, a terceira e quarta parte, chamadas de Teste Matemático e Teste de Memória, respectivamente, também são sub-processos do Protótipo Desempenho, que informa ordem de testes matemáticos e o outro a ordem de teste de memória. Os sub-processos estão demonstrados mais detalhadamente no português estruturado.

8.2 PORTUGUÊS ESTRUTURADO

O português estruturado demonstra a parte lógica do protótipo. O procedimento Executar é responsável pela execução dos testes. Este procedimento vai ser o primeiro a ser executado, chamando as funções Teste_Video, responsável pelo testes de Vídeo, Teste_Matemático, responsável pelos testes matemáticos e o Teste_Memoria, responsável pelo testes de memória.

Cada uma destas funções resulta no tempo em segundos que cada um deles leva para ser executado. Para uma melhor análise cada uma destas funções

foi subdividida em partes menores também resultando no tempo em segundos, que é impresso na tela para um acompanhamento passo a passo dos resultados. O português estruturado esta disponível no Apêndice 2.

8.3 AVALIAÇÃO DAS LINGUAGENS

8.3.1 Legibilidade

- Simplicidade Global

  O PocketStudio, além dos componentes básicos herdados do Pascal, necessita em muito do uso da API do Palm para muitas operações, o que acarreta em uma grande dificuldade para a legibilidade, pois o programador necessita saber a API do Palm além dos comandos próprios do PocketStudio.

  O Visual Basic .NET tem seus componentes básicos herdados do Visual Basic 6, mas que sofreram bastantes alterações depois que passou a utilizar a tecnologia .NET, considerando que muitos dos comandos utilizados durante o desenvolvimento não são do Visual Basic e sim do .NET, o que obriga o desenvolvedor ter conhecimento nas duas ferramentas. Claro que isto pode se tornar uma vantagem, caso todas as linguagens passem a usar a tecnologia .NET, uma vez que estes comandos vão ser comuns para estas linguagens. O .NET possibilita *overload* de operadores, que é a atribuição de um novo significado para os operadores +,-,/,*,AND,OR, entre outros, trazendo risco para a legibilidade, caso não seja usado de forma justificada.

A SuperWaba é muito semelhante ao Java, o que não traz nenhuma dificuldade para um programador Java passar a usá-lo. Usa apenas comandos próprios disponíveis em sua biblioteca e por isto não exige o conhecimento de outras tecnologias.

- Ortogonalidade

  Todas as linguagens avaliadas possuem um conjunto de instruções primitivas que possibilitam a criação de novos tipos de variáveis, criação de *arrays* e registros, sendo que no registro criado podem ser colocados vários atributos de vários tipos, inclusive outros registros ou tipos de variáveis criadas.

- Instruções de Controle

  Das três linguagens comparadas o SuperWaba é a única que não suporta o comando *goto*. Sendo este um comando que não deve ser usado, pois dificulta em muito a legibilidade do programa.

- Tipos de Dados e Estruturas

  Sobre o tipo de dados e estruturas, as linguagens avaliadas dão suporte a tipos de variáveis (inteiro, real, data, entre outros), inclusive o tipo lógico, que tira necessidade de utilizar variáveis inteiras para esta finalidade, sendo que o zero é considerado falso, e o um verdadeiro.

  Também as linguagens dão suporte à criação de estruturas de dados (*records*) para que as informações possam ser melhores armazenadas do que utilizar várias variáveis separadas.

- Considerações Sobre a Sintaxe

  O PocketStudio, como é baseado no Pascal tem alguns problemas na sintaxe, como por exemplo o controle de bloco de códigos, cujo o

finalizador é sempre *end*, não identificando assim qual bloco está sendo fechado. Inclusive no comando *repeat* o *end* não é utilizado o que deixa a sintaxe fora do restante do padrão da linguagem.

Em compensação o Visual Basic .NET possui bons finalizadores de blocos, sendo de fácil identificação no caso de *if, for, while.*

Já o SuperWaba, como ele utiliza a linguagem Java, todos os blocos de código são identificados com abre e fecha chaves, o que prejudica a sintaxe.

Nas três linguagens é impedido o uso de palavras reservadas como nome de variáveis ou funções, o que ajuda na legibilidade.

Em todas as linguagens avaliadas podem ser usados nomes das variáveis grandes para ajudar na identificação de cada um dos conteúdos.

8.3.2 Capacidade de Escrita

- Simplicidade e Ortogonalidade

  Todas as linguagens avaliadas possuem um conjunto completo de instruções primitivas, como *if, for, case, while, records*, criação de novos tipos de variáveis, que através destas podem dar origem a estruturas mais complexas. O PocketStudio tem limite no tratamento de *string*, devendo ser usado *array* de *char*, trazendo dificuldades ao trabalhar com variáveis *strings*, como por exemplo, ao somar duas *strings*, que deve ser usada função de concatenação.

- Suporte a Abstração

As três linguagens avaliadas têm suporte à abstração, ou seja, possibilidade de criar funções. Mas somente o SuperWaba e o Visual Basic .NET dão suporte a orientação a objetos, um item importante na abstração atualmente.

- Expressividade

  Tanto o SuperWaba, quanto o Visual Basic .NET, tem uma boa expressividade, pois permite juntar vários comandos de códigos uns com os outros de forma a construir uma estrutura enxuta.

  No entanto o PocketStudio não possibilita uma boa integração entre um comando e outro, necessitando de variáveis auxiliares e de mais linhas de código, não permitindo escrever um código simples. Logo abaixo é possível verificar um exemplo, onde foram necessárias cinco linhas de código.

```
StrIToA( Aux, Teste_Video );
StrCopy( Buffer, 'Tempo Teste VDO: ' );
StrCat( Buffer, Aux );
StrCat( Buffer, ' Segundo(s)' );
PSLabel.SetCaption( LblMediaTeste_VDO, Buffer );
```

8.3.3 Confiabilidade

- Verificação de Tipos

  Todas as linguagens fazem verificação de tipos e de erros em tempo de desenvolvimento. Para o Pocketstudio estas verificações são feitas no momento da compilação, já o SuperWaba e o Visual Basic .NET fazem a

verificação de erros durante a escrita do programa e inclusive dão sugestões de como resolver os erros.

Tanto Superwaba quanto PocketStudio resultam em um valor inteiro a divisão de dois números inteiros, que na verdade, pode dar um resultado do tipo real, e nem chegam a sugerir em forma de uma dica que isto pode trazer problemas em tempo de execução, dificultando na solução do problema. Para resolver este problema é necessário especificar que a divisão vai ser realizada entre números reais.

- Manipulação de Exceções

  Das três linguagens avaliadas, somente o PocketStudio não dá suporte ao tratamento de exceções, prejudicando na estabilidade do programa desenvolvido.

  Um caso que pode vir a tornar-se um problema é quando ocorre um estouro da capacidade de uma variável do tipo real, pois tanto o SuperWaba como no PocketStudio colocam conteúdo zero na variável. Apenas o Visual Basic .NET gera uma exceção (Erro). Quando a exceção não é gerada nesta situação é problemático, pois, fica difícil descobrir o que está ocorrendo, uma vez que o valor que deveria ser retornado está sendo retornado errado.

- Apelido (*aliasing*)

  Este problema pode ocorrer normalmente em 2 situações: na utilização de ponteiros e na utilização de passagem de variáveis por referência. Na utilização de ponteiros somente o PocketStudio pode apresentar este problema, já que tanto o SuperWaba, como o Visual Basic .NET não suportam que sejam utilizados ponteiros. Quanto à utilização de variáveis

por referência, a três linguagens avaliadas podem apresentar problemas se o recurso foi utilizado de forma errada.

A plataforma .NET possibilita a utilização de *weak reference*, que é um recurso onde uma classe pode apontar para outra classe depois de instanciada, onde pode vir a ocorrer um problema de Apelido.

Estes recursos são muito úteis, mas devem ser usados com prudência.

- Legibilidade e Capacidade de Escrita

  O PocketStudio utiliza muitas linhas para fazer um código simples, como mostra o exemplo abaixo.

```
Tempo_VDO := Teste_Video;
StrIToA( Aux, Tempo_VDO );
StrCopy( Buffer, 'Tempo Teste VDO: ' );
StrCat( Buffer, Aux );
StrCat( Buffer, ' Segundo(s)' );
PSLabel.SetCaption( LblMediaTeste_VDO, Buffer );
```

  Em compensação, este mesmo código acima pode facilmente ser escrito em apenas duas linha de código, tanto no SuperWaba, quanto no Visual Basic .NET. Como mostra o exemplo abaixo em Visual Basic .NET e SuperWaba:

  - Visual Basic .NET

```
Tempo_VDO = Teste_Video()
LblTeste_VDO.Text = "Tempo Teste VDO: “
+ Convert.ToString(Tempo_VDO) + " Segundo(s)“
```

  - SuperWaba

```
Tempo_VDO = Teste_Video();
```

```
lblTeste_VDO.setText( "Tempo Teste VDO: "
+ String.valueOf( Tempo_VDO ) + " Segundo(s)" );
```

A IDE do SuperWaba, como do Visual Basic .NET possibilita uma digitação de código rápido e correto, pois na chamada de qualquer método ou controle da linguagem, a própria IDE vai sugerindo o que deve ser escrito, evitando que seja escrito de forma errada e evitando a necessidade de decorar cada um dos métodos disponíveis na linguagem.

8.3.4 Custo

- Custo do Treinamento de Programadores

  Das três linguagens avaliadas a que tem mais centro de treinamento no país é o Visual Basic. NET, o que possibilita um custo menor, já que a concorrência é maior, enquanto o SuperWaba e PocketStudio somente possuem centro de treinamento na cidade de São Paulo. O SuperWaba ainda possui a opção de levar o instrutor a qualquer região do pais, desde que quem solicitar o treinamento, tenha uma estrutura física para este treinamento.

  O Visual Basic .NET é também a linguagem que possui maior número de material para os autodidata, pois possui muitos livros e muito material disponível na internet. Já o SuperWaba, possui uma boa documentação, mas fica restrito a isto e aos exemplos disponível no site do fabricante. Já o PocketStudio é a linguagem que tem menos material disponível na internet e fica restrito aos exemplos que acompanham a ferramenta, já que a documentação é muito fraca e em muitas vezes precisa ter

conhecimento da API do Palm, que também tem uma documentação simples.

- Custo de Escrita

  As linguagens SuperWaba e o Visual Basic .NET, possibilitam uma escrita objetiva e ainda são apoiadas por uma ótima IDE que vai indicando erros enquanto a pessoa digita o código e ainda tem a opção de auto completar, tirando necessidade do programador ter que conhecer como escrever cada comando ou método disponível na linguagem. Para comandos, classes e funções, que necessitam da passagem de parâmetros, já mostra quais os parâmetros devem ser passados. Em compensação, o PocketStudio não possibilita nenhum desses recursos, o que dificulta muito a aprendizagem e a programação. Mesmo estas características não estarem relacionadas diretamente com a linguagem e sim com a IDE utilizada, mas vale destacar que a IDE esta relacionada diretamente com a linguagem e vai influenciar no custo de escrita.

- Custo de Compilação

  Pelos computadores utilizados atualmente, compilar usando qualquer uma das linguagens avaliadas é rápido e simples, sem a necessidade de altos investimentos.

- Custo de Execução

  Pelo que foi avaliado com o desenvolvimento do programa de desempenho foi possível verificar que SuperWaba tem uma execução bem mais lenta, tanto no Pocket PC quanto no Palm, se comparado com o PocketStudio e Visual Basic .NET. Esta execução pode necessitar de um

equipamento mais potente para utilizar um programa desenvolvido nesta linguagem.

- Custo do Sistema e da Implementação da Linguagem

  Das linguagens avaliadas, todas funcionam com hardware de baixo custo, se comparado com os de top de linha. O PocketStudio tem grande vantagem neste quesito, pois pode desenvolver aplicativos compatíveis inclusive no PalmOS versão 2, possibilitando a utilização de equipamentos bem antigos. Segundo a documentação do SuperWaba, os aplicativos desenvolvidos com a sua biblioteca funcionam em todos os dispositivos PalmOS a partir da versão 3.0, no entanto, no programa de testes desenvolvido não funcionou na versão do PalmOS 3.5.

  A maquina virtual do SuperWaba necessita de aproximadamente 1MB de espaço livre no PDA, para a instalação tanto no Pocket PC, como no Palm, que para um dispositivo de pouca memória é um tamanho considerável.

- Custo da Má Confiabilidade

  Das três linguagens avaliadas, foi descoberto um *bug* na linguagem PocketStudio que pode trazer problemas no aplicativo após ter sido desenvolvido. Este *bug* está relacionado ao uso da instrução MOD com números acima de 10000, onde um erro de divisão por zero ocorre, finalizando o aplicativo.

  O SuperWaba também pode não ser confiável caso seja necessário executar na versão 5.5 da máquina virtual, um aplicativo desenvolvido para a versão 5.0 da máquina virtual.

- Custo de Manutenção dos Programas

Como o SuperWaba e o Visual Basic .NET suportam orientação a objetos, deixam a manutenção muito mais fácil, caso o aplicativo desenvolvido tenha usado este conceito. O PocketStudio não suporta orientação a objetos e por isto é um pouco prejudicado. Outro ponto, sobre o PocketStudio, é que ele utiliza muito a API do Palm, de forma que o programador se obriga a conhecer muito da API, para poder dar manutenção. Também deve ser considerado o fato de que a IDE do PocketStudio não traz informações importantes no meio do código, como por exemplo o tipo de variável que foi declarada, quais são os parâmetros e retornos utilizados numa função.

O SuperWaba, como utiliza uma máquina virtual, apresentou problemas entre versões da VM. Como por exemplo: o aplicativo de Desempenho que foi desenvolvido para a versão 5.0 não funcionou corretamente ao executar na versão 5.5 da VM. Isto gera problemas de compatibilidade entre aplicativos desenvolvidos em versões diferentes dificultando a manutenção.

8.4 DOCUMENTAÇÃO

A documentação do Visual Basic .NET é muito boa, pois conta com um *help* completo integrado com a ferramenta explicando cada método da plataforma, inclusive com exemplos e ainda conta com o auxilio do site MSDN, da Microsoft. Também existe uma grande variedade de livros que podem auxiliar o programador.

O SuperWaba contém uma documentação que acompanha o SDK, que contém todos os métodos disponíveis. Um ponto fraco dessa documentação, é que ela conta com poucos exemplos. Mas em compensação possui muitos aplicativos

prontos e com fontes disponíveis no site do fabricante que podem auxiliar o programador.

A documentação que acompanha o PocketStudio é muito simples e não possui muitos exemplos. Falta contemplar com um help sobre a API do sistema operacional PalmOS, que é muito requisitado dentro do código.

8.5 DIFICULDADES ENCONTRADAS NO DESENVOLVIMENTO

A seguir está a relação de limitações encontradas em cada uma das linguagens durante o desenvolvimento do protótipo.

8.5.1 PocketStudio

- Não pode criar nenhum objeto em tempo de execução (como um formulário), a menos que se use a API do Palm. Por não ter esta possibilidade, todos os componentes visuais devem ser colocados na tela em tempo de *designer*, o que dificulta a organização de alguns componentes na tela.
- Limite de 32*kbytes* no tamanho de variáveis.

  Esta limitação impossibilita em algumas operações que necessitem armazenar muitas informações no aparelho. Este limite de 32*kbytes* é para o aplicativo como um todo e não somente para uma função.
- Erro de Divisão por 0 na instrução MOD.

  Quando a instrução MOD é usada com um número superior a 100.000 apresenta um erro de divisão por zero. Como com valores menores o erro não acontecia, pode-se deduzir que este problema é uma falha da ferramenta.

- Funciona integrado apenas com o emulador da versão 4 do PalmOS.

  A versão atual do PalmOS é a 5.4, mas o emulador desta versão é incompatível com esta ferramenta de desenvolvimento. Isto impossibilita utilizar o *debug* numa aplicação desenvolvida que utiliza recursos exclusivos desta versão do PalmOS.

- Limitação na resolução de apenas 160X160 pixels.

  Não é possível desenvolver aplicativos na resolução superior a 160X160 pixels. A maioria dos Palm novos está disponível na resolução 320X320 pixels ou 480X320 pixels.

8.5.2 SuperWaba

- Complexidade na instalação no Pocket PC.

  A VM precisa estar obrigatoriamente instalada no Pocket PC na pasta Raiz:\SuperWaba\ e o aplicativo precisa estar na pasta Raiz:\SuperWaba\[Nome do Aplicativo]\, pois caso esse critério não seja obedecido o aplicativo não funciona.

- Tratamento específico entre plataformas

  Ao apagar arquivos via código de programação é preciso indicar o caminho completo (diretório) do arquivo no Pocket PC. Isto é ruim, pois não mantêm a compatibilidade de execução do aplicativo entre Pocket PC e o Palm, já que no Palm não existem diretórios e todos os arquivos ficam na pasta principal do equipamento.

- Complexidade na instalação do compilador

  A Instalação do Compilador no PC é complexa, pois exige a instalação de vários arquivos, como o J2RE da Sun, mais a biblioteca SDK do

Superwaba e um editor Java, no caso, o Eclipse. Também exige configuração das bibliotecas para cada novo projeto.

- Falta de um editor Visual para montar as telas.

  Como não existe um editor visual o trabalho para montar a interface do aplicativo é grande principalmente quando o mesmo aplicativo vai funcionar em diferentes resoluções de tela.

- Não possibilita acesso a API do equipamento

  Por se tratar de uma linguagem multiplataforma, esta não fornece acesso a API do dispositivo. Mas, por conseqüência, pode trazer algum tipo de limitação.

8.5.3 Visual Basic .NET

Durante todo o processo de desenvolvimento do protótipo não foram encontradas dificuldades.

A única limitação encontrada é que ele, por ser multi-plataforma, não possui acesso direto a API do dispositivo, o que pode limitar a utilização de recursos específicos do equipamento.

8.6 AVALIAÇÃO DOS RESULTADOS DO PROTÓTIPO

Conforme a metodologia definida no capítulo 7, foi desenvolvido um protótipo, para realização dos testes de desempenho da linguagem. Os equipamentos utilizados para esta análise foram: um equipamento utilizando o sistema operacional PalmOS e outro utilizando Windows CE. Estes equipamentos continham as seguintes características:

- PalmOS

  Palm Zire71

  Processador: Texas Instruments 144Mhz de Clock

  Memória: 16MB

  Vídeo: 320X320 pixels – 65mil cores

- Windows CE

  Pocket PC HP IPAQ 1940

  Processador: Samsung 266Mhz de Clock

  Memória: 64MB

  Vídeo: 320X240 pixels – 65mil cores

O quadro 14 mostra as linguagens de programação avaliadas e em qual sistema operacional cada linguagem foi testada. Logo abaixo da legenda há a quadro com o tempo obtido para executar cada um dos testes, considerando que o aplicativo desenvolvido em Superwaba foi executado tanto no Windows CE quanto no PalmOS, pois o mesmo é multiplataforma.

QUADRO 14: RESULTADO DOS TESTES DE DESEMPENHO - PARTE 1 - 2005

Legenda
PS =PocketStudio (PalmOS) – Pascal
VB =Visual Basic .NET (Pocket PC) – Basic
SW =SuperWaba (PalmOS/Pocket PC) – Java

Resultado Geral dos Testes

	VB - Pocket PC	PS - Palm	SW - Pocket PC	SW - PalmOS
VDO - Mudar Cores	15 Segundo(s)	5 Segundo(s)	14 Segundo(s)	13 Segundo(s)
VDO - Criar Objetos	10 Segundo(s)	2 Segundo(s)	14 Segundo(s)	21 Segundo(s)
MAT - Fatorial	1 Segundo(s)	9 Segundo(s)	1 Segundo(s)	34 Segundo(s)
MAT - Número PI	1 Segundo(s)	32 Segundo(s)	1 Segundo(s)	87 Segundo(s)
MAT - Números Primos	1 Segundo(s)	4 Segundo(s)	1 Segundo(s)	51 Segundo(s)
MEM - Gravar Arquivos	3 Segundo(s)	4 Segundo(s)	153 Segundo(s)	17 Segundo(s)
MEM - Ler Arquivos	19 Segundo(s)	1 Segundo(s)	3 Segundo(s)	12 Segundo(s)
Tempo Total	**50 Segundo(s)**	**57 Segundo(s)**	**187 Segundo(s)**	**235 Segundo(s)**

Comparação entre o Visual Basic .NET e SuperWaba no Windows CE

	VB - Pocket PC	SW - Pocket PC	Desempenho VB com relação ao SW
VDO - Mudar Cores	15 Segundo(s)	14 Segundo(s)	-6,67%
VDO - Criar Objetos	10 Segundo(s)	14 Segundo(s)	40,00%
MAT - Fatorial	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Número PI	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Números Primos	1 Segundo(s)	1 Segundo(s)	0,00%
MEM - Gravar Arquivos	3 Segundo(s)	153 Segundo(s)	5000,00%
MEM - Ler Arquivos	19 Segundo(s)	3 Segundo(s)	-84,21%
Tempo Total	**50 Segundo(s)**	**187 Segundo(s)**	**274,00%**

Comparação entre o PocketStudio e SuperWaba no PalmOS

	PS - Palm	SW - PalmOS	Desempenho PS com relação ao SW
VDO - Mudar Cores	4 Segundo(s)	13 Segundo(s)	225,00%
VDO - Criar Objetos	3 Segundo(s)	21 Segundo(s)	600,00%
MAT - Fatorial	9 Segundo(s)	34 Segundo(s)	277,78%
MAT - Número PI	32 Segundo(s)	87 Segundo(s)	171,88%
MAT - Números Primos	4 Segundo(s)	51 Segundo(s)	1175,00%
MEM - Gravar Arquivos	5 Segundo(s)	17 Segundo(s)	240,00%
MEM - Ler Arquivos	1 Segundo(s)	12 Segundo(s)	1100,00%
Tempo Total	**58 Segundo(s)**	**235 Segundo(s)**	**305,17%**

Na tabela de comparação entre o Visual Basic .NET e SuperWaba, pode-se observar que o Visual Basic .NET teve um desempenho muito parecido ao apresentado pelo SuperWaba. Apesar do quadro de resultados apresentar uma diferença de 5.000% no teste de gravação de arquivos, foi porque o SuperWaba gera um arquivo do tipo “.PDB”, que é nativo do PalmOS, para gravar os registros. Esta lentidão é reconhecida inclusive pelo fabricante do SuperWaba. Caso não ocorresse este problema a diferença seria bem menor.

Também foi observado que o Visual Basic .NET foi 84,21% mais lento que o SuperWaba no teste de leitura de arquivos, 6,67% mais lento no teste de mudar cores e 40% mais rápido no teste de criar objetos. Como no código escrito em cada uma das linguagens não houveram limitações, esta diferença de tempo é atribuída diretamente a cada uma das linguagens.

O SuperWaba é uma linguagem que gera um código intermediário para ser executado pela máquina virtual, enquanto o aplicativo vai sendo executado. O Visual Basic .NET também gera o código intermediário, mas é compilado por completo no momento da execução do programa pelo Compact Framework .NET. Pode-se concluir através deste teste que a metodologia utilizada pelo .NET traz um desempenho equivalente ao SuperWaba na maior parte dos casos.

Na comparação entre o PocketStudio e o SuperWaba na plataforma PalmOS foi possível perceber que a linguagem que gera um executável nativo da plataforma foi extremamente mais rápido, ganhando em todos os testes, chegando a mais de 305% de diferença no resultado geral.

Foi realizado um segundo experimento com os mesmos testes, porém trocando a função de cálculo do fatorial para uma função recursiva. Também foi observado que a função de cálculo do número PI não estava chegando a um

resultado correto. Este problema ocorria nas linguagens PocketStudio e SuperWaba na divisão de números inteiros, que retornava um número truncado ao invés de retornar um número de ponto flutuante. Para contornar este problema foi necessário transformar os números inteiros em números de ponto flutuante, para depois fazer a divisão. Outro item verificado foi que o PocketStudio utiliza somente uma resolução de 160X160 pixels da tela mesmo ela sendo maior e expande para preencher a tela inteira. Após esta verificação foi alterado o aplicativo desenvolvido nas outras duas linguagens para utilizar somente uma área de 160X160 pixels nos testes de vídeo, ficando equivalente ao PocketStudio. Durante a leitura de arquivos foi substituído um *array* que armazenava em memória os 2000 registros gravados, por uma variável que armazenava apenas 1 registro. Esta substituição do *array* pela variável foi necessário pois o PocketStudio possui um limite de apenas 32*kbytes* para o tamanho de variáveis, impossibilitando a declaração do *array*.

Ao fazer estas alterações foi observada uma diferença nos resultados, como mostra o gráfico 3 e 6, se comparado com o teste anterior. No quadro 15 é possível observar os novos resultados.

QUADRO 15: RESULTADO DOS TESTES DE DESEMPENHO - PARTE 2 - 2005

Legenda
PS =PocketStudio (PalmOS) – Pascal
VB =Visual Basic .NET (Pocket PC) – Basic
SW =SuperWaba (PalmOS/Pocket PC) – Java

Resultado Geral dos Testes

	VB - Pocket PC	PS - Palm	SW - Pocket PC	SW – PalmOS
VDO - Mudar Cores	8 Segundo(s)	5 Segundo(s)	10 Segundo(s)	13 Segundo(s)
VDO – Criar Objetos	8 Segundo(s)	2 Segundo(s)	11 Segundo(s)	20 Segundo(s)
MAT - Fatorial	1 Segundo(s)	0 Segundo(s)	1 Segundo(s)	78 Segundo(s)
MAT - Número PI	1 Segundo(s)	29 Segundo(s)	1 Segundo(s)	129 Segundo(s)
MAT - Números Primos	1 Segundo(s)	4 Segundo(s)	1 Segundo(s)	68 Segundo(s)
MEM - Gravar Arquivos	3 Segundo(s)	4 Segundo(s)	153 Segundo(s)	19 Segundo(s)
MEM - Ler Arquivos	19 Segundo(s)	1 Segundo(s)	1 Segundo(s)	12 Segundo(s)
Tempo Total	**41 Segundo(s)**	**45 Segundo(s)**	**180 Segundo(s)**	**339 Segundo(s)**

Comparação entre o Visual Basic .NET e SuperWaba no Windows CE

	VB - Pocket PC	SW - Pocket PC	Desempenho VB com relação ao SW
VDO - Mudar Cores	8 Segundo(s)	10 Segundo(s)	25,00%
VDO - Criar Objetos	8 Segundo(s)	11 Segundo(s)	37,50%
MAT - Fatorial	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Número PI	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Números Primos	1 Segundo(s)	1 Segundo(s)	0,00%
MEM - Gravar Arquivos	3 Segundo(s)	153 Segundo(s)	5000,00%
MEM - Ler Arquivos	19 Segundo(s)	1 Segundo(s)	-94,74%
Tempo Total	**41 Segundo(s)**	**180 Segundo(s)**	**339,02%**

Comparação entre o PocketStudio e SuperWaba no PalmOS

	PS - Palm	SW - PalmOS	Desempenho PS com relação ao SW
VDO - Mudar Cores	5 Segundo(s)	13 Segundo(s)	160,00%
VDO - Criar Objetos	2 Segundo(s)	20 Segundo(s)	900,00%
MAT - Fatorial	0 Segundo(s)	78 Segundo(s)	0,00%
MAT - Número PI	29 Segundo(s)	129 Segundo(s)	344,83%
MAT - Números Primos	4 Segundo(s)	68 Segundo(s)	1600,00%
MEM - Gravar Arquivos	4 Segundo(s)	19 Segundo(s)	375,00%
MEM - Ler Arquivos	1 Segundo(s)	12 Segundo(s)	1100,00%
Tempo Total	**45 Segundo(s)**	**339 Segundo(s)**	**653,33%**

O cálculo do fatorial através da recursividade fez com que no PalmOS ocorre diferença nos resultados, tanto no SuperWaba, quando no PocketStudio. O interessante desta alteração foi que o PocketStudio passou a executar este teste de forma bem mais rápida, enquanto o SuperWaba teve um aumento de mais de 100% na duração deste teste.

A alteração na resolução de vídeo para 160X160 pixels trouxe para as linguagens SuperWaba e Visual Basic .NET uma redução no tempo do teste de vídeo, mas apenas para o Windows CE, enquanto o SuperWaba com o PalmOS permaneceu igual, mostrando que a área na tela que está sendo atualizada pode trazer diferença de performance, mas somente no Windows CE, enquanto no PalmOS a tela deve ser atualizada por inteira, mesmo usando apenas uma parte dela.

Após modificar o cálculo do número PI, para realizar divisões que antes eram entre números inteiros, por números reais, trouxe a informação que o PocketStudio é prejudicado com estes tipos de cálculo, pois ficou mais lento, enquanto o SuperWaba passou a executar este teste de forma mais rápida, mostrando que esta linguagem oferece um melhor desempenho em cálculos usando números reais do que inteiros.

Com a substituição do *array* por uma variável simples durante a leitura dos registros, somente o SuperWaba executando sob o Windows CE trouxe uma redução de 66% no tempo de leitura, mas o restante permaneceu igual.

Os testes de gravar arquivos e números primos não foram alterados entre um teste e outro, mas apresentaram uma redução no tempo dos testes na linguagem SuperWaba rodando no PalmOS. E esta mudança de tempo deve estar

relacionada às outras alterações que foram realizadas e acabaram por influenciar no restante do teste. Este segundo realizado pode ser verificado no quadro 15.

Como estes testes foram realizados apenas com dois equipamentos, foi adicionado no Apêndice 3 alguns resultados obtidos com outros equipamentos, para que pudessem auxiliar na escolha da linguagem de programação.

Nos gráficos 1 e 2 é possível comparar melhor a diferença que ocorreu nos resultados da parte 1 e da parte 2 dos testes para a plataforma Pocket PC.

GRÁFICO 1: COMPARAÇÃO SUPERWABA – POCKET PC

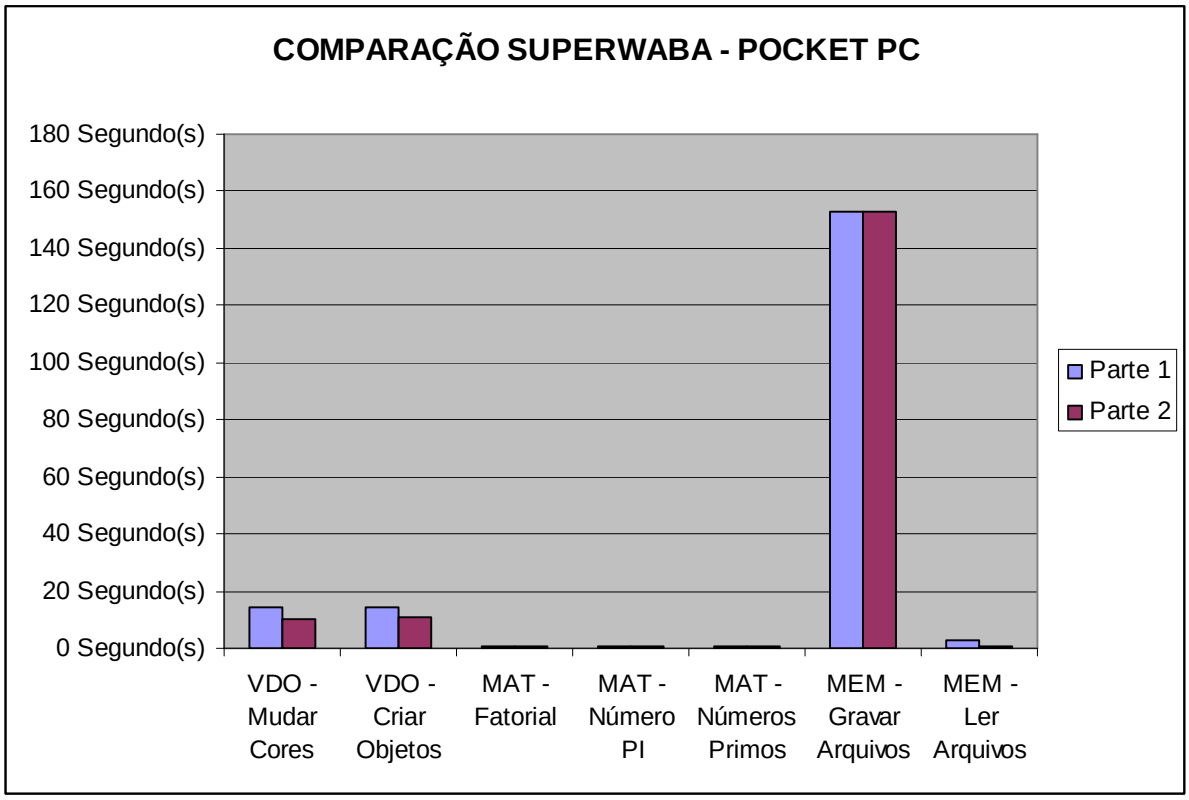
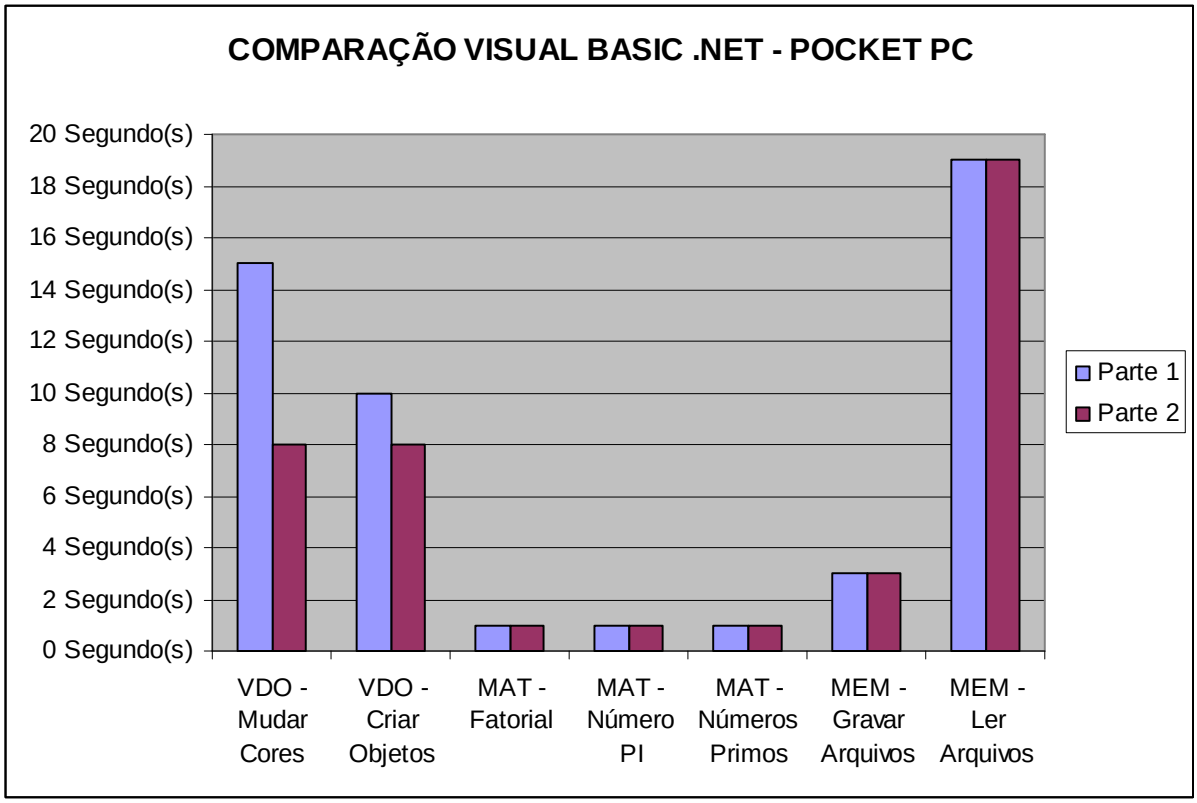
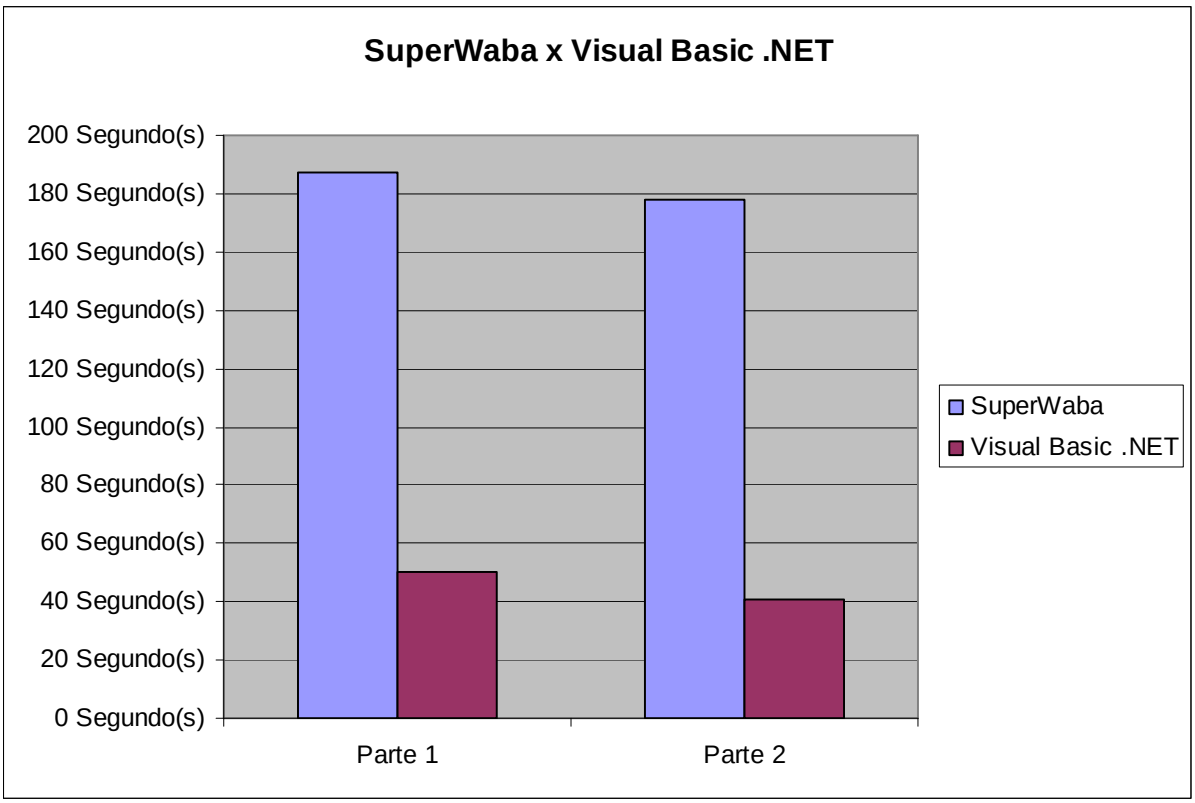


GRÁFICO 2: COMPARAÇÃO VISUAL BASIC .NET – POCKET PC



O gráfico 3 mostra a comparação entre o SuperWaba e VisualBasic .NET, onde é possível verificar facilmente que o SuperWaba é muito mais lento que o Visual Basic .NET, pois demorou quase 180 segundos para realizar todo o teste, enquanto o protótipo desenvolvido em Visual Basic .NET levou menos de 50 segundos para realizar todo o teste.

GRÁFICO 3: COMPARAÇÃO SUPERWABA x VISUAL BASIC .NET



Nos gráficos 4 e 5 é possível comparar melhor a diferença que ocorreu nos resultados da parte 1 e da parte 2 dos testes para a plataforma PalmOS.

GRÁFICO 4: COMPARAÇÃO SUPERWABA – PALMOS

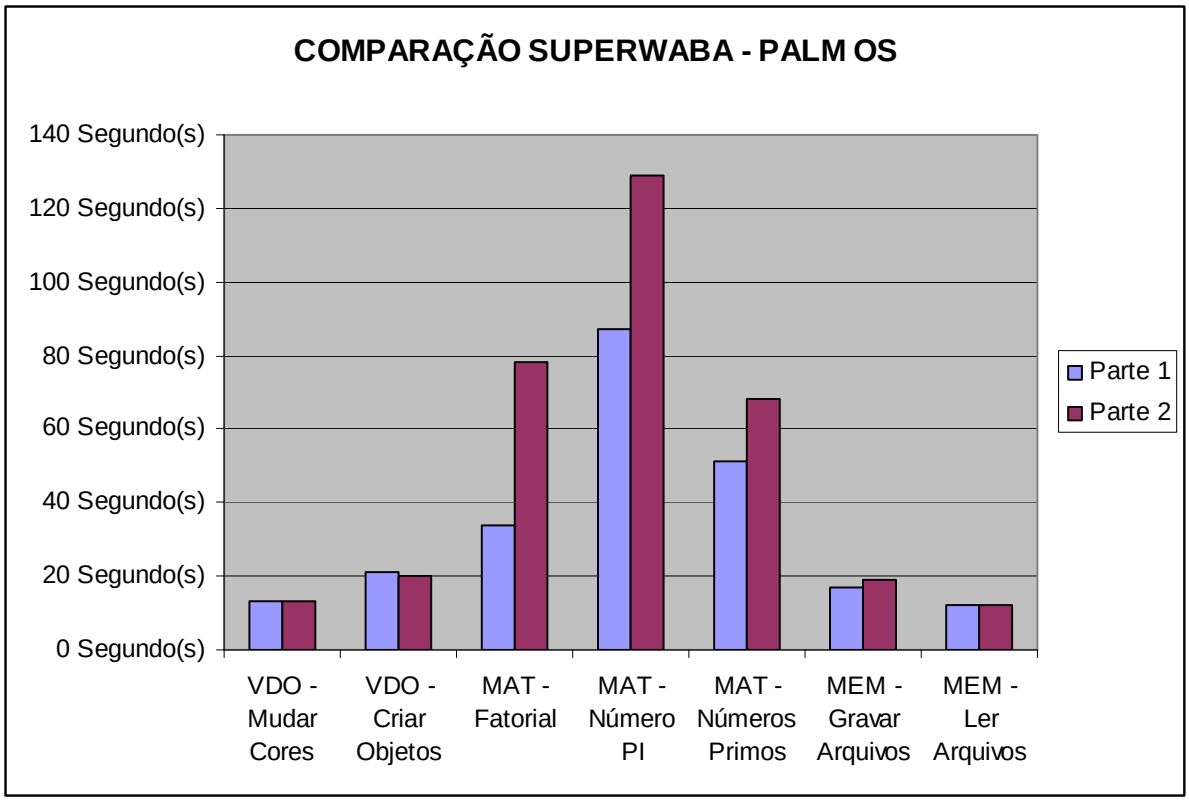
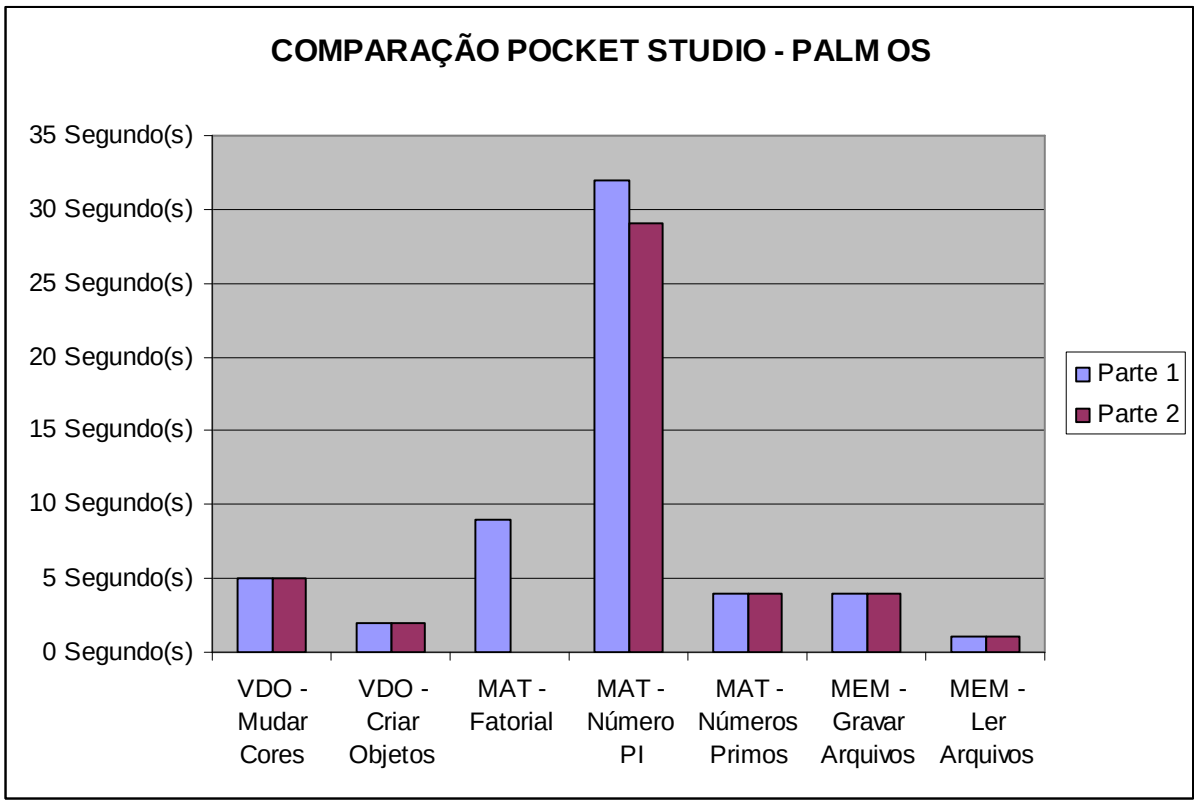
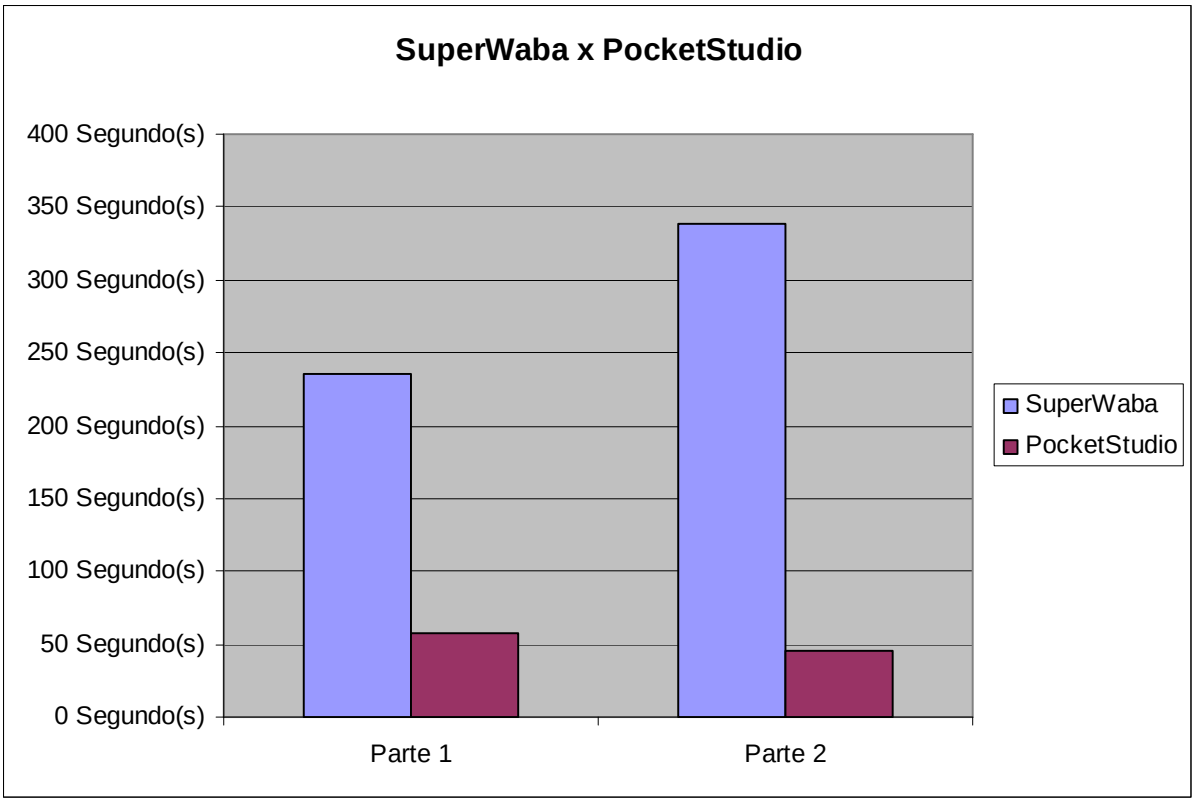


GRÁFICO 5: COMPARAÇÃO POCKETSTUDIO – PALM OS



O gráfico 6 mostra a comparação entre o SuperWaba e PocketStudio, onde é possível verificar facilmente que o SuperWaba é muito mais lento que o PocketStudio, pois demorou quase 350 segundos para realizar todo o segundo teste, enquanto o protótipo desenvolvido em PocketStudio levou menos de 50 segundos.

GRÁFICO 6: COMPARAÇÃO SUPERWABA x POCKETSTUDIO



9 CONCLUSÃO

Nunca os dispositivos portáteis foram tão poderosos, oferecendo aos usuários novas e fascinantes formas de comunicação e de administração para seus negócios. Manter a informação acessível em qualquer lugar é uma necessidade cada vez mais constante e em muitos casos simplesmente a internet não resolve o problema. Para suprir essa necessidade, a utilização de dispositivos móveis com aplicativo desenvolvido de forma específica para atender cada situação é uma boa solução, pois traz ao usuário a informação da maneira que ele precisa. Mas para que isto seja possível é necessário a utilização de uma linguagem de desenvolvimento para dispositivos móveis.

Para ajudar na escolha da linguagem de programação, esta monografia teve como objetivo comparar linguagens para dispositivos móveis e criar um protótipo para realizar testes de desempenho entre linguagens. A análise dos resultados obtidos trouxe informações importantes para escolha de uma linguagem de programação que irá suprir as necessidades do programador conforme a área para qual o software será desenvolvido.

Dentre as linguagens avaliadas, que foram o SuperWaba, PocketStudio e Visual Basic .NET, e que tiveram um protótipo desenvolvido nesta monografia, evidenciou-se que estas linguagens possuem muitas diferenças umas das outras, tanto na questão de sintaxe como também no desempenho. Foi possível verificar grande diferença de desempenho, onde o SuperWaba se mostrou mais lento que as linguagens nativas, principalmente em cálculos matemáticas no caso do Palm OS e na gravação de arquivos no Pocket PC, mas em compensação possibilita a portabilidade. Também mostrou que uma simples alteração de código gera

diferenças consideráveis nos resultados finais, como foi o caso do cálculo do fatorial que passou a ser recursiva e teve resultados bastante interessante.

Elas possuem uma forma de desenvolvimento muito próximo a de uma linguagem para PC, exceto que deve-se prestar muita atenção no desenvolvimento do aplicativo para não desperdiçar recursos, pelo fato dos equipamentos terem poder de processamento limitado, se comparando a um PC.

A evolução tecnológica em geral, e a computação móvel em particular, apesar de terem assumido muita importância nas atividades profissionais ainda não são exploradas de forma completa, a ponto de usar todos os recursos essas eles podem oferecer.

Para maior mobilidade e agilidade nas atividades desenvolvidas do dia a dia, é imprescindível á escolha de uma linguagem de programação para desenvolver soluções inteligentes que venham a integrar processos e trazer para junto do usuário informações atualizadas.

Para trabalhos futuros, pode-se avaliar outras linguagens de programação seguindo o mesmo método usado nesta monografia de forma a trazer mais opções de escolha para o programador.

Com o surgimento da tecnologia. NET, que promete portabilidade, poderá também ser avaliado o desempenho da linguagem Visual Basic. NET, tendo o aplicativo compilado rodando no PalmOS, caso venha a existir um compact .NET framework para este sistema operacional.

Entre os sistemas operacionais de dispositivos móveis avaliados, pode-se incluir o sistema operacional Sybiam, já que a linguagem SuperWaba disponibiliza uma máquina virtual para o mesmo.

O protótipo pode ser incrementado para salvar os testes realizados e comparar com uma base pré-existente, pois além de comparar as linguagens, vai passar a comparar equipamentos, ajudando na escolha para aquisição de um equipamento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEXANDRONI, Márcio. *Desenvolvimento Palm com PocketStudio*. Touch Informática Ltda, 2001.

CODE Lines. Disponível em: http://www.codelines.com/. Acessado em: 26 de março de 2005.

CRAIG, John Clark, *Microsoft Visual Basic: Versao3*. Tradução e revisão técnica e adaptação dos programas Jeremias René Descartes P. dos Santos. São Paulo: Editora Markon Books, 1994.

FRANKLIN, Keith, *VB,NET para desenvolvedores*. Tradução Aldir José Coelho Corrêa da Silva, Revisão Técnica: Roberto Gomes de Aguiar Veiga. São Paulo: Editora Markon Books, 2002.

AHO, Alfred V., SETHI, Ravi e ULLMAN, Jeffrey D., *Compiladores – Princípios, Técnicas e Ferramentas*. Tradução Daniel de Ariosto Pinto. Rio de Janeiro: Editora Guanabara Koogan S.A., 1995.

FUNDÃO da Computação. Disponível em: http://www.fundao.pro.br/. Acessado em: 14 de maio de 2005.

GERBER, Luciano. *Disciplina de Paradigmas de Linguagens de Programação*. Disponível em: http://www.ulbra.tche.br/~lgerber/Paradigmas/Paradigmas_2aAula_16032000.pdf. Acessado em 28 de março de 2005.

GRUPO Palm Br. Disponível em: http://www.palm-br.com.br. Acessado em 22 de janeiro de 2005.

HAZAN, Guilherme Campos. *SuperWaba* Disponível em: http://www.superwaba.com.br. Acessado em: 25 de maio de 2005.

KNUDSEN , Jonathan e NIEMEYER, Patrick. *Aprendendo Java*. Editora Campus, 2000.

LOMAX, Paul, PETRUSKA, Ron e ROMAN, Steven. *Linguagem VB .NET*. 1 ed. – Editora Campus, 2002.

MICROSOFT Windows Mobile. Disponível em: http://www.microsoft.com/windowsmobile/. Acessado em: 21 abril de 2005.

MSDN. Disponível em http://msdn.microsoft.com. Acessado em: dia 26 de março de 2005.

PALM Brasil. Disponível em: http://www.palmbrasil.com.br. Acessado em 17 de abril de 2005.

PALM is now PalmOne and PalmSource. Disponível em: http://www.palm.com. Acessado em: 24 de abril de 2005.

PALM Land. Disponível em: http://www.palmland.com.br. Acessado em 12 de março de 2005.

POCKET Studio. Disponível em: http://www.winsoft.sk/pstudio.htm. Acessado em 14 de maio 2005.

RITCHEY, Tim. *Programando com Java! Beta 2.*Tradução Geraldo Costa Filho. Rio de Janeiro: Editora Campus, 1997.

SCHÜTZER , Waldeck e MASSAGO, Sadao. *Java*, 1999 Disponível em: http://www2.dm.ufscar.br/%7Ewaldeck/curso/java/. Acessado em 25 de maio de 2005.

SEBESTA, Robert W. *Conceitos de linguagens de programação.* Tradução José Carlos Barbosa dos Santos. – 5. ed. – Porto Alegre: Editora Bookman, 2003.

SUN Microsystems. Disponível em: http://www.sun.com/java/. Acessado em 14 de maio de 2005.

TROIS, Julio. *PalmTop para Iniciantes e Experts.* Editora VisualBooks, 2003.

WAKEFIELD, Cameron, SONDER, Henk-Evert e LEE, Wei Meng. *VB .NET – Guia do Desenvolvedor*. 1. ed – Editora Atlas Books, 2002.

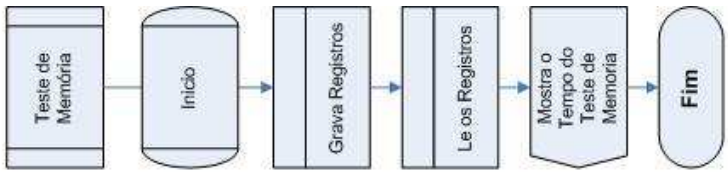
WIKIPEDIA. Disponível em: http://pt.wikipedia.org/wiki/Compilador Acessado em 12 de março de 2005.

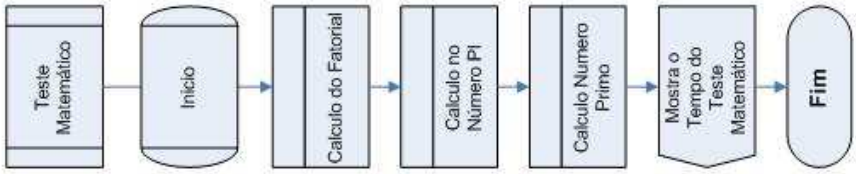
WIKIPEDIA. Disponível em: http://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o Acessado em 12 de março de 2005.

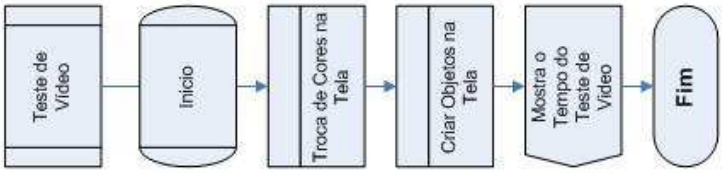
WIKIPEDIA. Disponível em: http://pt.wikipedia.org/wiki/Interpretadores Acessado em 12 de março de 2005.

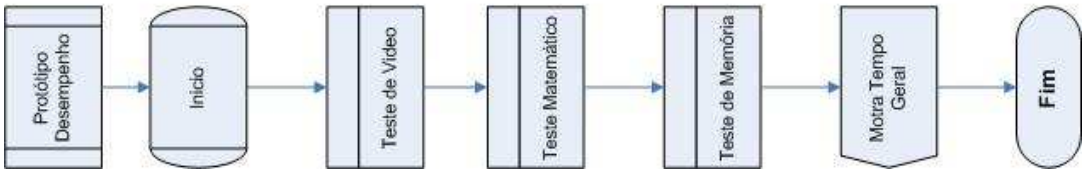
WINCE Brasil. Disponível em: http://www.wince.com.br/. Acessado em: 22 de abril de 2005.

APÊNDICE 1 – FLUXOGRAMA









APÊNDICE 2 – PORTUGUÊS ESTRUTURADO

```
programa Desempenho
Declaracoes


Inicio

  procedimento Executar
  Declaracoes
    Inteiro Tempo_VDO, Tempo_MAT, Tempo_MEM, Tempo_Total

  Inicio

    Tempo_VDO <- Teste_Video
    Escreva( "Tempo Teste VDO: " + Tempo_VDO + " Segundo(s)" )

    Tempo_MAT <- Teste_Matematico
    Escreva( "Tempo Teste MAT: " + Tempo_MAT + " Segundo(s)" )

    Tempo_MEM <- Teste_Memoria
    Escreva( "Tempo Teste VDO: " + Tempo_MEM + " Segundo(s)" )

    Tempo_Total <-Tempo_VDO + Tempo_MAT + Tempo_MEM
    Escreva( "Tempo Geral: " + Tempo_Total + " Segundo(s)" )

  Fim

  Inteiro funcao Teste_Matemetico
  Declaracoes
    Inteiro Teste_1, Teste_2, Teste_3, Total

  Inicio

    Teste_1 <- MAT_Fatorial
    Escreva( "Fatorial: " + Teste_1 + " Segundo(s)" )

    Teste_2 <- MAT_NumeroPI
    Escreva( "Número PI: " + Teste_2 + " Segundo(s)" )

    Teste_3 <- MAT_NumerosPrimos
    Escreva( "Números Primos: " + Teste_3 + " Segundo(s)" )

    Total <- Teste_1 + Teste_2 + Teste_3

    Retorna( Total )

  Fim
```

```
Inteiro funcao Teste_Video
Declaracoes
  Inteiro Teste_1, Teste_2, Total

Inicio

  Teste_1 <- VDO_MudarCores
  Escreva( "Mudar Cores: " + Teste_1 + " Segundo(s)" )

  Teste_2 <- VDO_CriarObjetos
  Escreva( "Criar Objetos: " + Teste_2 + " Segundo(s)" )

  Total <- Teste_1 + Teste_2

  Retorna( Total )

Fim

Inteiro funcao Teste_Memoria
Declaracoes
  Inteiro Teste_1, Teste_2, Total

Inicio

  Teste_1 <- MEM_GravarArquivos
  Escreva( "Gravar Arquivos: " + Teste_1 + " Segundo(s)" )

  Teste_2 <- MEM_LerArquivos
  Escreva( "Ler Arquivos: " + Teste_2 + " Segundo(s)" )

  Total <- Teste_1 + Teste_2

  Retorna( Total )

Fim


Inteiro funcao MAT_Fatorial
Declaracoes
  Inteiro X, FAT
  Hora Hora_Inicio, Hora_Fim

Inicio

  Hora_Inicio <- HoraAtual

  para X <- 1 ate 70000 passo 1

    FAT = Fatorial( 10 )

  fimpara
```

```
  Hora_Fim <- HoraAtual

  Retorna( Hora_Fim - Hora_Inicio )

Fim

Inteiro funcao Fatorial( Inteiro N )
Inicio

  se( N > 1 ) entao
    Retorna( N * Fatorial( N - 1 )
  senao
    Retorna( 1 )

Fim


Inteiro funcao MAT_NumeroPI
Declaracoes
  Hora Hora_Inicio, Hora_Fim
  Real A, B, PI
  Inteiro I, SIGN

Inicio

  Hora_Inicio <- HoraAtual

  A <- 16/5
  B <- 4/239
  PI <- A - B

  SIGN <- -1

  I <- 3
  enquanto( I < 700000 )

    A <- A/25
    B <- B/57121

    PI <- PI + SIGN * ( A - B ) / I
    SIGN <- - SIGN

    I <- I + 2

  fimenquanto

  Hora_Fim <- HoraAtual

  Retorna( Hora_Fim - Hora_Inicio )
```

```
Fim


Inteiro funcao MAT_NumerosPrimos
Declaracoes
  Inteiro Prox_Divisor, Qtd_Divisores
  Hora Hora_Inicio, Hora_Fim
  Logico Resultado

Inicio

  Hora_Inicio <- HoraAtual

  Prox_Divisor <- 1
  Qtd_Divisores <- 2

  enquanto( Prox_Divisor <= 700000 )

    se( 700000 mod Prox_Divisor = 0 ) entao
      Qtd_Divisores <- Qtd_Divisores + 1
    fimse

    Prox_Divisor <- Prox_Divisor + 1

  fimenquanto

  se( Qtd_Divisores = 2 ) entao
    Resultado <- Verdadeiro
  senao
    Resultado <- Falso
  fimse

  Hora_Fim <- HoraAtual

  Retorna( Hora_Fim - Hora_Inicio )

Fim

Inteiro funcao VDO_MudarCores
Declaracoes
  Hora Hora_Inicio, Hora_Fim
  Inteiro I, R, G, B

Inicio

  Hora_Inicio <- HoraAtual

  para I <- 0 ate 3000 passo 1

    R <- Randomico( 255 )
```

```
    G <- Randomico( 255 )
    B <- Randomico( 255 )

    DesenharQuadrado( 160, 160, Cor(R,G,B) )

  fimpara

  Hora_Fim <- HoraAtual

  Retorna( Hora_Fim - Hora_Inicio )

Fim

Inteiro funcao VDO_CriarObjetos
Declaracoes
  Hora Hora_Inicio, Hora_Fim
  Inteiro I, X, Y, R, G, B, Raio, scLargura, scAltura

Inicio

  Hora_Inicio <- HoraAtual

  scLargura <- 160
  scAltura <- 160

  Raio <- 17
  X <- Raio
  Y <- Raio

  para I <- 0 ate 3000 passo 1

    R <- Randomico( 255 )
    G <- Randomico( 255 )
    B <- Randomico( 255 )

    DesenharCirculo( 160, 160, Cor(R,G,B) )

    se( X < scLargura - Raio ) entao
      X <- X + 5
    senao

      X <- Raio

      se( Y < scAltura - Raio ) entao
        Y <- Y + 5
      senao
        Y <- Raio
      fimse


    fimse
```

```
    fimpara

    Hora_Fim <- HoraAtual

    Retorna( Hora_Fim - Hora_Inicio )

  Fim

  Inteiro funcao MEM_GravarArquivos
  Declaracoes
    Hora Hora_Inicio, Hora_Fim
    Inteiro I

  Inicio

    Hora_Inicio <- HoraAtual

    CriarArquivo( "Desempenho" )

    para I <- 0 ate 2000 passo 1

      EscrevaArquivo( "Projeto de Graduação - Teste de
Desempenho" )

    fimpara

    Hora_Fim <- HoraAtual

    Retorna( Hora_Fim - Hora_Inicio )

  Fim

  Inteiro funcao MEM_LerArquivos
  Declaracoes
    Hora Hora_Inicio, Hora_Fim
    Caracter Texto
    Inteiro I

  Inicio

    Hora_Inicio <- HoraAtual

    AbrirArquivo( "Desempenho" )

    para I <- 0 ate 2000 passo 1

      Texto <- LeiaArquivo( I )

    fimpara
```

```
    ApagarArquivo( "Desempenho" )
    Hora_Fim <- HoraAtual
    Retorna( Hora_Fim - Hora_Inicio )
  Fim

Fim
```

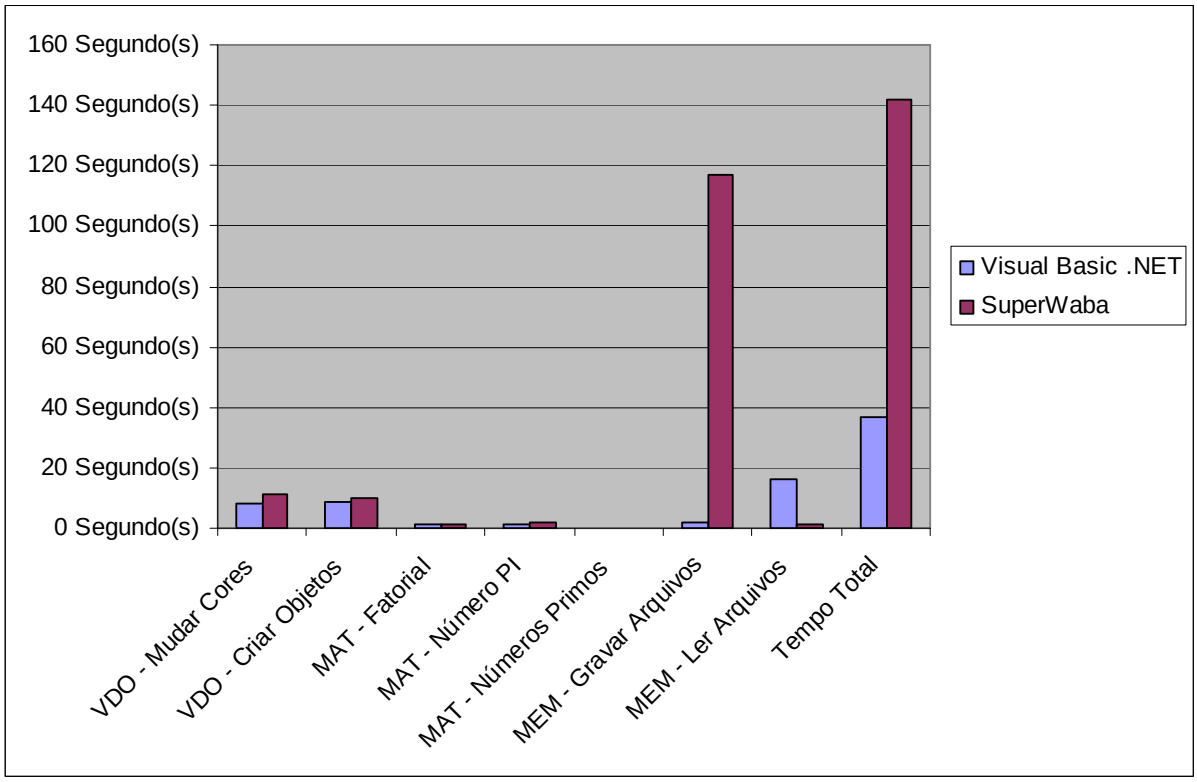
APÊNDICE 3 – RESULTADOS COM OUTROS EQUIPAMENTOS

HP IPAQ - RX 3715

Processador: Samsung S3C 400 Mhz de clock
Resolução da Tela: 240x320 pixels
Sistema Operacional: Microsoft Mobile Pocket PC 2003 SE
Memória: 152 MB

Legenda
VB =Visual Basic .NET (Windows CE) - Basic
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	Visual Basic .NET	SuperWaba	Comparação
VDO - Mudar Cores	8 Segundo(s)	11 Segundo(s)	37,50%
VDO - Criar Objetos	9 Segundo(s)	10 Segundo(s)	11,11%
MAT - Fatorial	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Número PI	1 Segundo(s)	2 Segundo(s)	100,00%
MAT - Números Primos	0 Segundo(s)	0 Segundo(s)	0,00%
MEM - Gravar Arquivos	2 Segundo(s)	117 Segundo(s)	5750,00%
MEM - Ler Arquivos	16 Segundo(s)	1 Segundo(s)	-93,75%
Tempo Total	**37 Segundo(s)**	**142 Segundo(s)**	**283,78%**

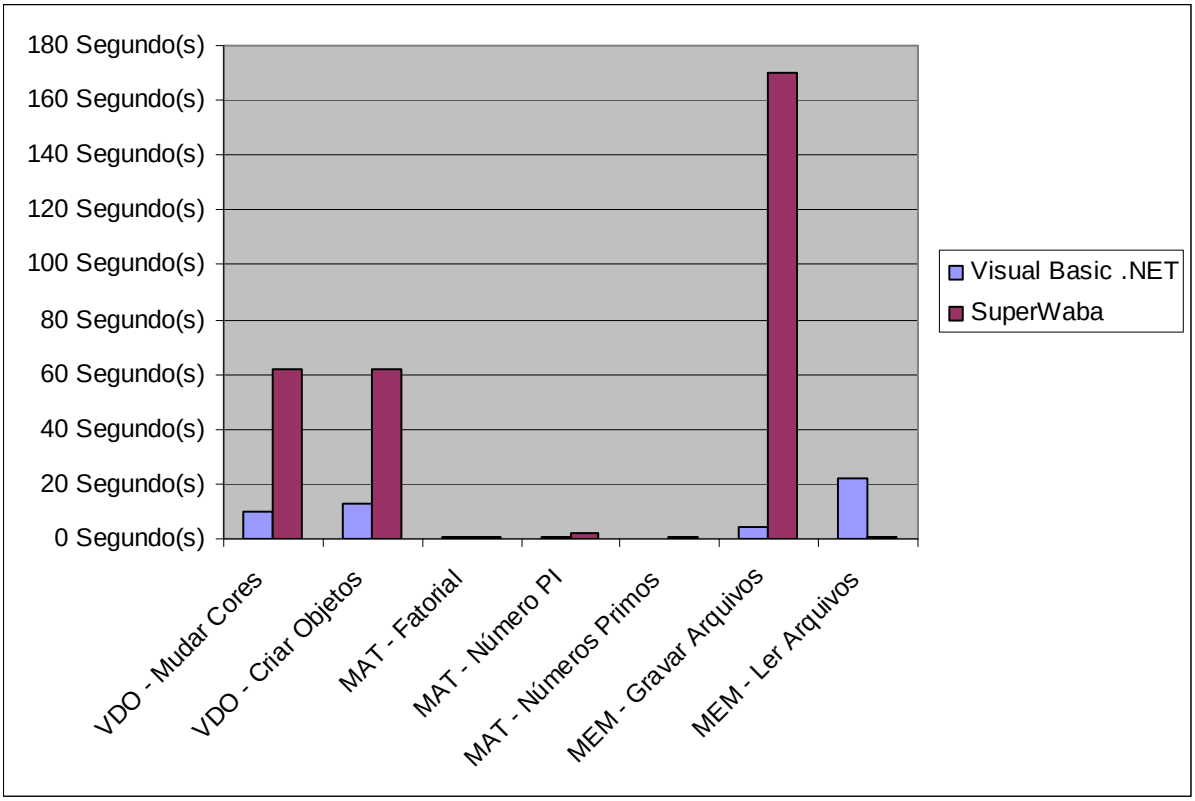


HP IPAQ – 2210

Processador: Intel XScale 400 Mhz de clock
Resolução da Tela: 240x320 pixels
Sistema Operacional: Microsoft Mobile Pocket PC 2003 SE
Memória: 64 MB

Legenda
VB =Visual Basic .NET (Windows CE) - Basic
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	Visual Basic .NET	SuperWaba	Comparação
VDO - Mudar Cores	10 Segundo(s)	62 Segundo(s)	520,00%
VDO - Criar Objetos	13 Segundo(s)	62 Segundo(s)	376,92%
MAT - Fatorial	1 Segundo(s)	1 Segundo(s)	0,00%
MAT - Número PI	1 Segundo(s)	2 Segundo(s)	100,00%
MAT - Números Primos	0 Segundo(s)	1 Segundo(s)	0,00%
MEM - Gravar Arquivos	4 Segundo(s)	170 Segundo(s)	4150,00%
MEM - Ler Arquivos	22 Segundo(s)	1 Segundo(s)	-95,45%
Tempo Total	**51 Segundo(s)**	**299 Segundo(s)**	**486,27%**

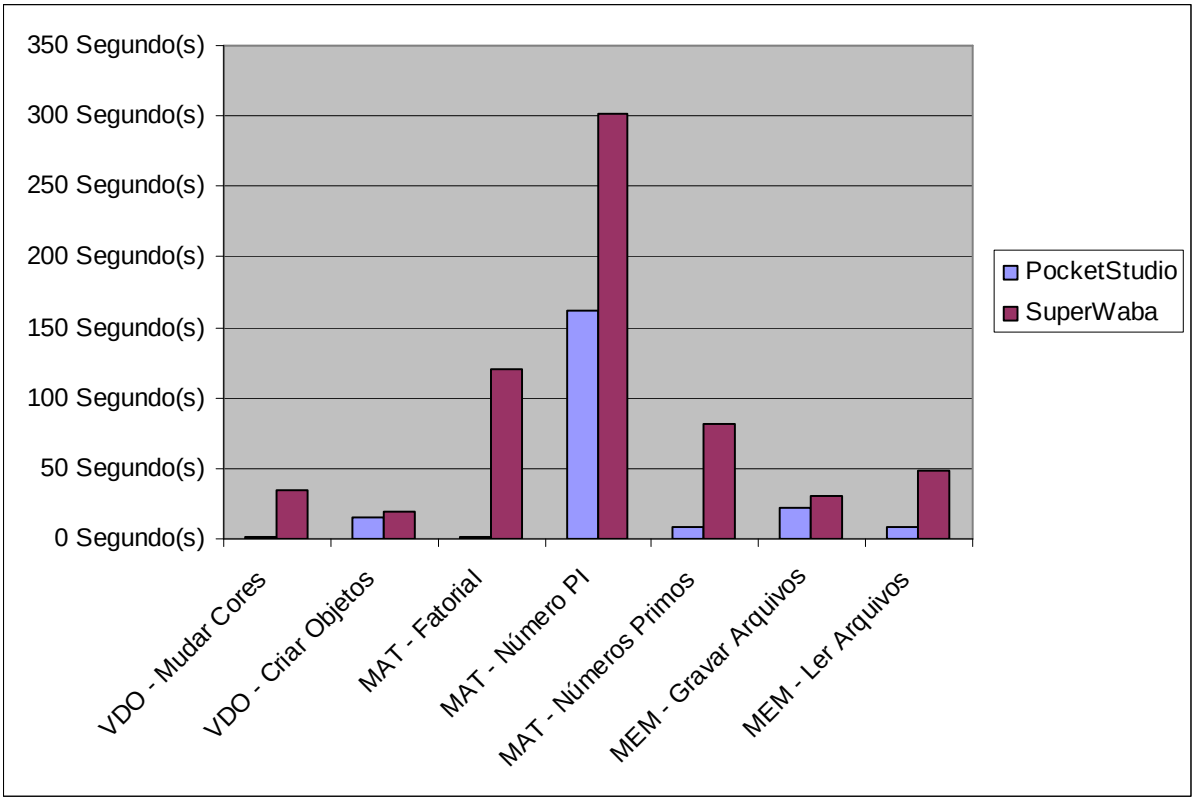


Palm M130

Processador: Motorola Dragonball VZ 33 Mhz de clock
Resolução da Tela: 160x160 pixels
Sistema Operacional: Palm OS 4.1
Memória: 8MB

Legenda
PS =Pocket Studio (PalmOS) - Pascal
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	PocketStudio	SuperWaba	Comparação
VDO - Mudar Cores	2 Segundo(s)	34 Segundo(s)	1600,00%
VDO - Criar Objetos	15 Segundo(s)	19 Segundo(s)	26,67%
MAT - Fatorial	1 Segundo(s)	121 Segundo(s)	12000,00%
MAT - Número PI	162 Segundo(s)	301 Segundo(s)	85,80%
MAT - Números Primos	8 Segundo(s)	81 Segundo(s)	912,50%
MEM - Gravar Arquivos	22 Segundo(s)	31 Segundo(s)	40,91%
MEM - Ler Arquivos	8 Segundo(s)	49 Segundo(s)	512,50%
Tempo Total	**218 Segundo(s)**	**636 Segundo(s)**	**191,74%**

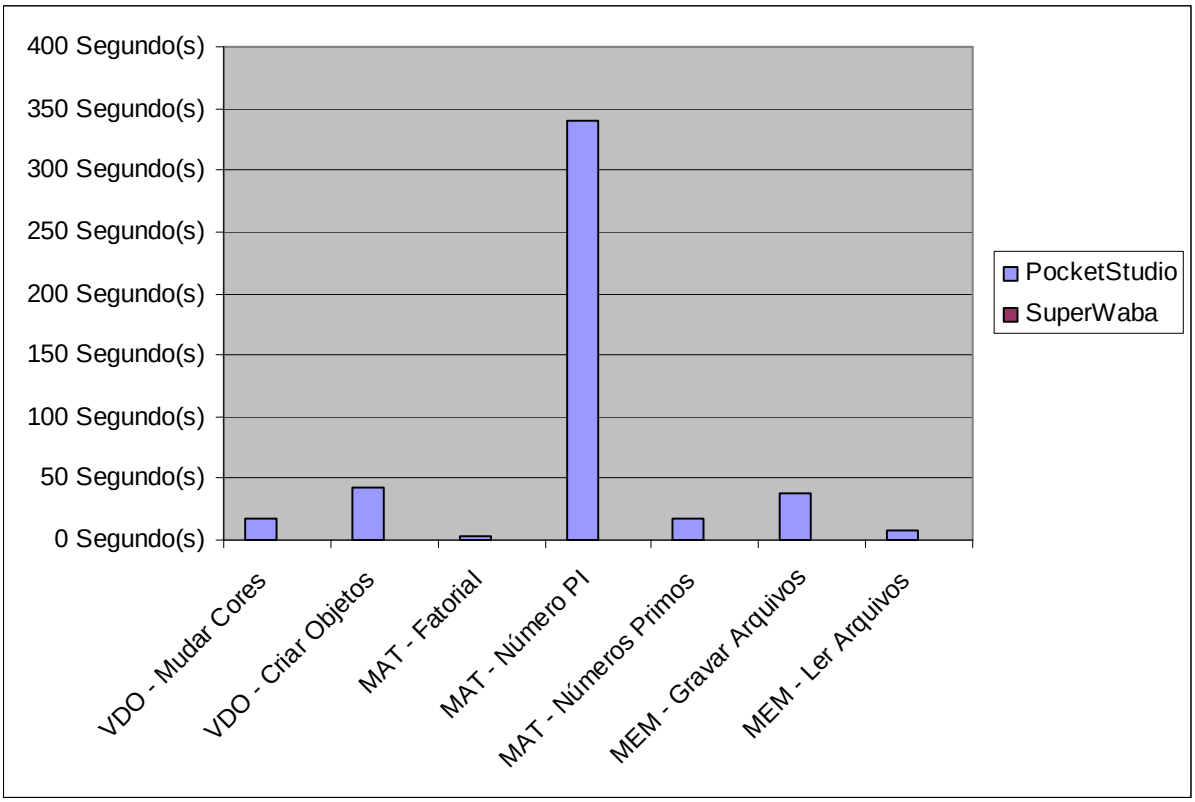


Palm M100

Processador: Motorola Dragonball EZ 16 MHz de clock
Resolução da Tela: 160x160 pixels
Sistema Operacional: Palm OS 3.5
Memória: 2 MB

Legenda
PS =Pocket Studio (PalmOS) - Pascal
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	PocketStudio	SuperWaba	Comparação
VDO - Mudar Cores	18 Segundo(s)	0 Segundo(s)	-100,00%
VDO - Criar Objetos	43 Segundo(s)	0 Segundo(s)	-100,00%
MAT - Fatorial	3 Segundo(s)	0 Segundo(s)	-100,00%
MAT - Número PI	340 Segundo(s)	0 Segundo(s)	-100,00%
MAT - Números Primos	18 Segundo(s)	0 Segundo(s)	-100,00%
MEM - Gravar Arquivos	38 Segundo(s)	0 Segundo(s)	-100,00%
MEM - Ler Arquivos	8 Segundo(s)	0 Segundo(s)	-100,00%
Tempo Total	**468 Segundo(s)**	**0 Segundo(s)**	**-100,00%**



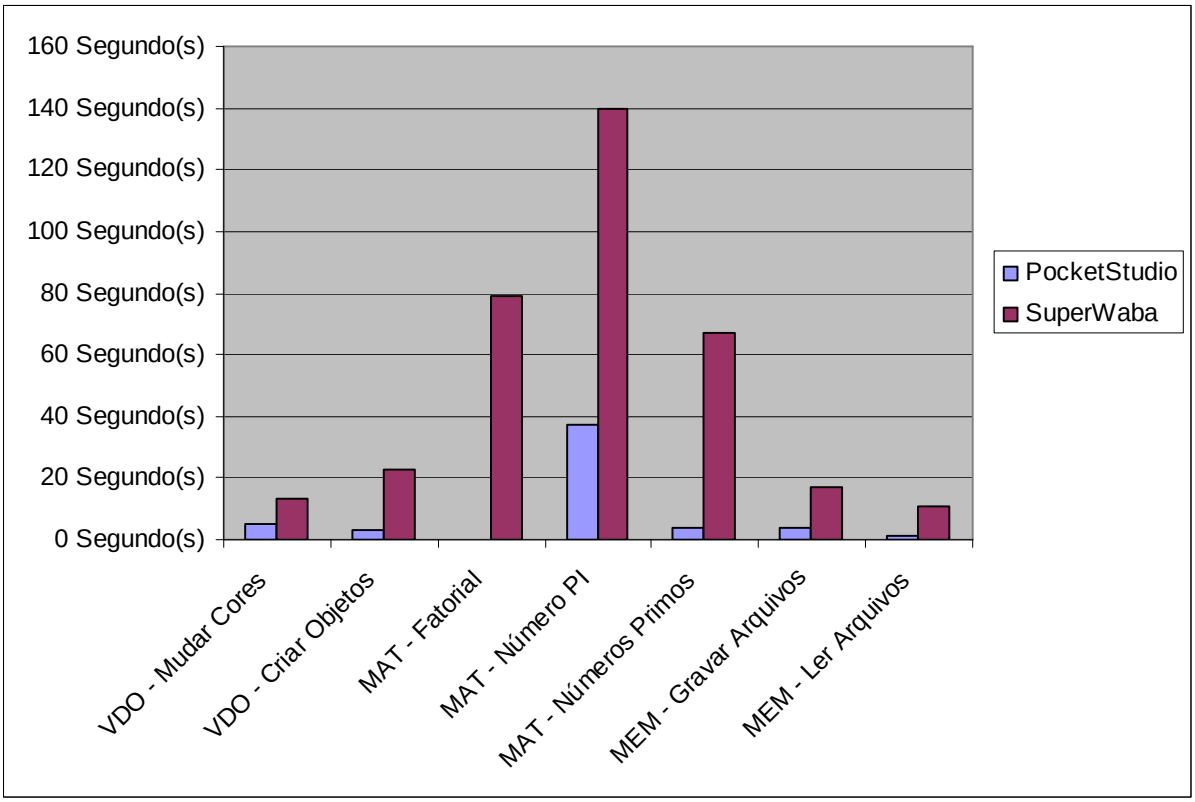
Obs.: Durante a execução do SuperWaba, ocorreu o seguinte erro:
ERRO: systemMgr.c, Line:156,Unimplemented

Palm TE

Processador: Texas Instruments 126 MHz de clock
Resolução da Tela: 320x320 pixels
Sistema Operacional: Palm OS 5.2.1
Memória: 32 MB

Legenda
PS =Pocket Studio (PalmOS) - Pascal
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	PocketStudio	SuperWaba	Comparação
VDO - Mudar Cores	5 Segundo(s)	13 Segundo(s)	160,00%
VDO - Criar Objetos	3 Segundo(s)	23 Segundo(s)	666,67%
MAT - Fatorial	0 Segundo(s)	79 Segundo(s)	0,00%
MAT - Número PI	37 Segundo(s)	140 Segundo(s)	278,38%
MAT - Numeros Primos	4 Segundo(s)	67 Segundo(s)	1575,00%
MEM - Gravar Arquivos	4 Segundo(s)	17 Segundo(s)	325,00%
MEM - Ler Arquivos	1 Segundo(s)	11 Segundo(s)	1000,00%
Tempo Total	**54 Segundo(s)**	**350 Segundo(s)**	**548,15%**

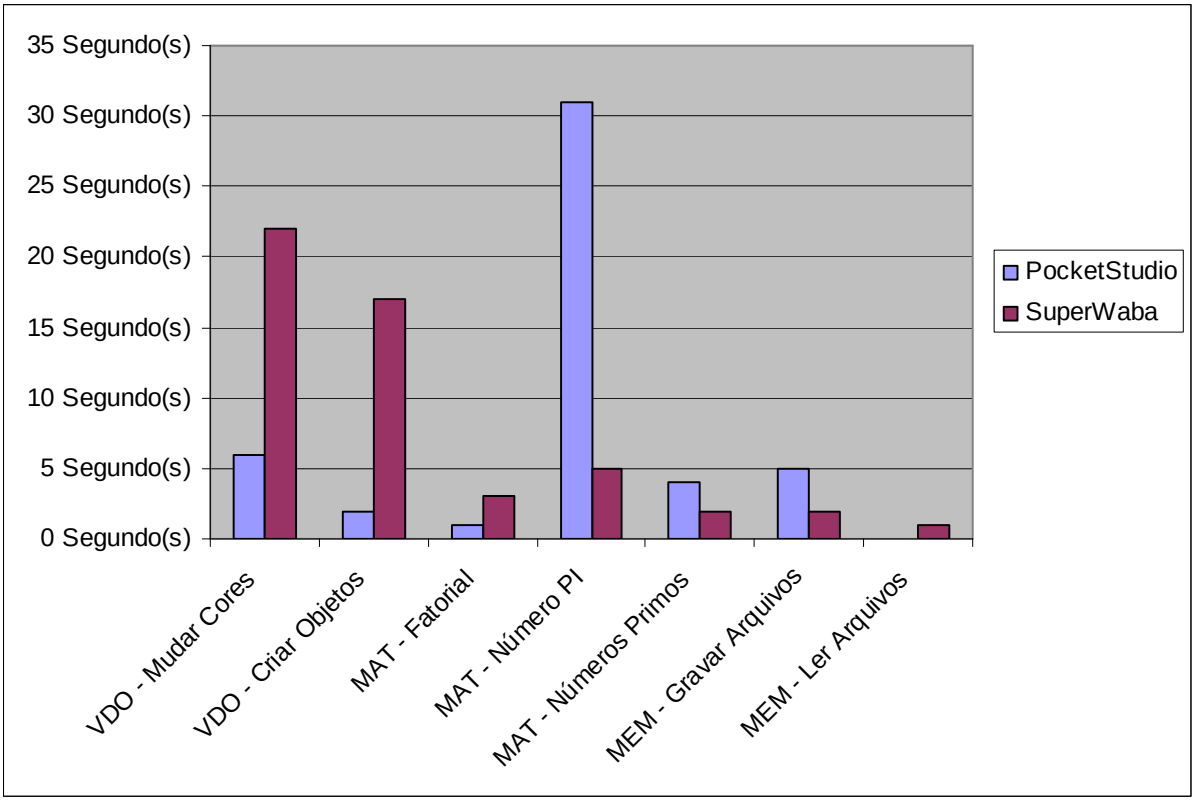


Palm TT

Processador: Texas Instruments 144 Mhz de clock
Resolução da Tela: 320x320 pixels
Sistema Operacional: Palm OS v5.0
Memória: 16MB

Legenda
PS =Pocket Studio (PalmOS) - Pascal
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	PocketStudio	SuperWaba	Comparação
VDO - Mudar Cores	6 Segundo(s)	22 Segundo(s)	266,67%
VDO - Criar Objetos	2 Segundo(s)	17 Segundo(s)	750,00%
MAT - Fatorial	1 Segundo(s)	3 Segundo(s)	200,00%
MAT - Número PI	31 Segundo(s)	5 Segundo(s)	-83,87%
MAT - Numeros Primos	4 Segundo(s)	2 Segundo(s)	-50,00%
MEM - Gravar Arquivos	5 Segundo(s)	2 Segundo(s)	-60,00%
MEM - Ler Arquivos	0 Segundo(s)	1 Segundo(s)	0,00%
Tempo Total	**49 Segundo(s)**	**52 Segundo(s)**	**6,12%**



Palm Zire21

Processador: Texas Intruments 126 MHz de clock
Resolução da Tela: 160x160 pixels
Sistema Operacional: Palm OS 5.2.1
Memória: 8MB

Legenda
PS =Pocket Studio (PalmOS) - Pascal
SW =SuperWaba (PalmOS/Windows CE) - Java

Resultado Geral dos Testes			
	PocketStudio	SuperWaba	Comparação
VDO - Mudar Cores	1 Segundo(s)	10 Segundo(s)	900,00%
VDO - Criar Objetos	1 Segundo(s)	23 Segundo(s)	2200,00%
MAT - Fatorial	1 Segundo(s)	71 Segundo(s)	7000,00%
MAT - Número PI	32 Segundo(s)	127 Segundo(s)	296,88%
MAT - Numeros Primos	4 Segundo(s)	68 Segundo(s)	1600,00%
MEM - Gravar Arquivos	4 Segundo(s)	17 Segundo(s)	325,00%
MEM - Ler Arquivos	1 Segundo(s)	11 Segundo(s)	1000,00%
Tempo Total	**44 Segundo(s)**	**327 Segundo(s)**	**643,18%**