Artigo: Evolução de Software

Autor: Douglas José Peixoto de Azevedo

1 INTRODUÇÃO

O *software* tornou-se o elemento chave da evolução dos sistemas e produtos baseados em computadores [Som92]. Nos últimos tempos ele evolui de uma ferramenta de análise de informações e de resolução de problemas especializados para uma industria da programação. Mas, logo, a cultura e a história criaram um conjunto de problemas que persiste até hoje. Será que o *software* tornou-se um fator limitante na evolução dos sistemas?

O *software* de computador é uma informação que existe em duas formas básicas[Pre95]: componentes não executáveis em máquina e componentes executáveis em máquina. Os componentes são criados por meio de uma série de conversões que mapeam as exigências do cliente para código executável em máquina. Um modelo ou protótipo das exigências é convertido num projeto.

O projeto de *software* é convertido numa forma de linguagem que especifica a estrutura de seus dados, os atributos procedimentais e os requisitos relacionados. A forma de linguagem é processada por um tradutor que a converte em instruções executáveis em máquina.

A "reusabilidade"[Som92] é uma característica importante de um componente de alta qualidade, ou seja o componente deve ser projetado de forma que possa ser "reusado" em muitos programas diferentes. Na década de 1960, construíamos bibliotecas de sub-rotinas científicas que eram "reusáveis" num amplo conjunto de aplicações científicas e de engenharia. Essas bibliotecas de sub-rotinas "reusavam" algoritmos bem definidos efetivamente, mas tinham um domínio de aplicação limitado. Atualmente ampliamos nossa visão do "reuso" a fim de envolver não somente algoritmos, mas também estruturas de dados.

Um componente "reusável" da década de 1990 engloba tanto dados como processamento num único pacote (as vezes chamado classe ou objeto), possibilitando ao engenheiro de *software* crie novas aplicações a partir de partes "reusáveis". Como exemplo podemos citar: as interfaces interativas de hoje que freqüentemente, são construídas utilizando-se componentes "reusáveis" que possibilitam a criação de janelas gráficas, menus *pull-down* e uma ampla variedade de mecanismos de interação. As estruturas de dados e detalhes de processamento exigidos para se construir a interface com os usuários estão contidas numa biblioteca de componentes "reusáveis" para construção de interfaces.

Os componentes[Sho83] de *software* são construídos usando uma linguagem de programação que tem um vocabulário limitado, uma gramática explicitamente definida e regras de sintaxe e semântica bem formadas. Esses atributos são essenciais para a tradução por máquina. As formas de linguagem em uso são linguagens de máquina, linguagem de alto nível e linguagens não procedimentais. A Linguagem de máquina é uma representação simbólica do conjunto de instruções da unidade central de Processamento (CPU). As linguagens de alto nível permitem que o desenvolvedor de *software* e o programa sejam independentes da máquina. No decorrer da última década, um grupo de linguagens de quarta geração ou não procedimentais foi introduzido. Em vez de exigir que o desenvolvedor de *software* especifique detalhes de procedimentos, a linguagem não procedimental subentende um programa especificando o resultado desejado em vez de especificar a ação exigida

para se conseguir esse resultado. O *software* de apoio converte a especificação do resultado num programa executável em máquina.

Este trabalho não quer polemizar o assunto, mas discutir as visões dos pesquisadores referentes ao assunto. Para isto, está dividido da seguinte forma: no item 2 deste trabalho, mostraremos conceitos e recursos de *software;* categorias de software são abordadas no item 3; no item 4, mostramos o ciclo de vida clássico; no item 5, uma visão do papel evolutivo do *software*.

2 SOFTWARE

2.1 Conceitos e Recursos

Verdadeiramente notável nos computadores não é a alta velocidade nem a incrível capacidade de armazenamento de dados, mas sim, a variedade ilimitada de usos a quem pode atender.

O que torna isso possível, com um número limitado de componentes diferentes, é o *software*[Ver84], [Pre95]. É importante entender que o mais sofisticado *hardware* de computador do mundo não é importante sozinho. Ligue-o na tomada e tudo que ele pode fazer é zumbir com o som característico dos aparelhos eletrônicos. Qualquer atitude que ele tome a partir daí deve ser explicitamente ordenada através de instruções. Essas instruções são escritas por desenvolvedores e são organizadas de forma seqüencialmente lógica. Apenas após estas instruções estarem devidamente organizadas é que os componentes eletrônicos do sistema de computação podem realizar suas tarefas na seqüência correta para atingir os seus objetivos. As instruções processáveis por computadores são chamadas de *software* porque os desenvolvedores podem facilmente mudá-las, ao contrário do *hardware* onde as modificações são mais difíceis. Em um certo sentido, o computador é muito parecido com um automóvel, que é fabricado com alguns componentes que podem fazer muitas coisas, mas que só efetivamente as realizam quando o motorista o controla através dos vários dispositivos de interface (a direção, o pedal do acelerador e a alavanca de câmbio). O motorista opera como o *software,* fornecendo ao automóvel uma seqüência específica de instruções que o levará ao seu destino. Durante a viagem o motorista pode ter que alterar a seqüência de instruções a fim de se acomodar às condições do tráfego, aos semáforos e aos pedestres. Da mesma maneira o *software* dos computadores permite que a seqüência de instruções seja alterada enquanto está sendo executada pelo computador.

O *software* está organizado em programas que são mantidos geralmente nos grandes bancos de dados das organizações (bibliotecas), fazendo com que possam ser rapidamente acessados pelo sistema, quando necessários. Arquivos especiais chamados programas de biblioteca são mantidos para o armazenamento dos programas, tão logo eles são escritos pelos desenvolvedores.

3 CATEGORIAS DE SOFTWARE

Os desenvolvedores gastam bastante tempo para projetar os passos precisos e lógicos que devem formar um sistema de processamento de dados.

Verzello[ver84] classifica o *software* em três tipos, sendo:

- *Software* de sistema - são programas escritos para controlar e coordenar o *software*.
- *Software* de linguagens - são programas que traduzem outros programas escritos em linguagens de programação mais ou menos semelhantes à língua inglesa, para a forma binária que é a linguagem utilizada pelos componentes do sistema computacional e, além disso, os programas escritos para ajudar os desenvolvedores a escrever seus programas e a manter os programas já escritos a salvo, em bancos de dados especiais.

- *Software* de aplicação - são programas escritos para resolver problemas comerciais ou prestar outros serviços de processamento de dados aos usuários.

- Já Pressman[Pre95], amplia esta classificação de *software* para 7 categorias, comentando ser uma tarefa um tanto difícil desenvolver categorias genéricas para aplicações de *software*, pois à medida em que a complexidade do *software* cresce, desaparece a clara visão em compartimentos:
- *Software* básico - é uma coleção de programas escritos para dar apoio a outros programas. A área do *software* básico é caracterizada por: forte interação com o hardware de computador; intenso uso por múltiplos usuários; operações concorrentes que exigem escalonamento "*schedule*"; compartilhamento de recursos e sofisticada administração do processo; estruturas de dados complexas e múltiplas interfaces externas.
- *Software* de tempo real - monitora, analisa e controla eventos do mundo real. Entre os elementos do *software* de tempo real incluem-se: um componente de coleta de dados que obtém as informações provenientes de um ambiente externo, um componente de análise que transforma as informações conforme a aplicação exige; um componente de controle / saída que responde ao ambiente externo e um componente de monitoração que coordena todos os demais componentes de forma a resposta em tempo real. O termo "tempo real" difere de "interativo" ou "*time sharing*" (tempo compartilhado). Um sistema de tempo real deve responder dentro de restrições de tempo estritas. O tempo de resposta de um sistema interativo pode ser normalmente ultrapassado sem resultados desastrosos.
- *Software* comercial - é a maior área particular de *software*. As aplicações dessa área reestruturam os dados de uma forma que facilita as operações comerciais e as tomadas de decisões administrativas. Além da aplicação de processamento de dados convencional, as aplicações de *software* comerciais abrangem a computação interativa.
- *Software* científico e de engenharia - tem sido caraterizado por algoritmos de processamento de números. As aplicações variam da astronomia à vulcanologia da análise de fadiga mecânica de automóveis, à dinâmica orbital de naves espaciais recuperáveis e da biologia molecular à manufatura automatizada.
- *Software* embutido - é usado para controlar produtos e sistemas para os mercados industriais e de consumo. O *software* embutido ("*embedded software*") reside na memória só de leitura "*read only*" e pode executar funções limitadas e particulares (por exemplo, controle de teclado para fornos de microondas) ou oferecer recursos funcionais de controle significativos (por exemplo, funções digitais em automóveis, tais como controle, mostradores no painel, sistemas de freio, etc.)
- *Software* de computador pessoal - são os *softwares* para computadores pessoais que entrou em efervescência na última década, tais como processamento de textos, planilhas eletrônicas, computação gráfica, diversões, gerenciamento de dados, aplicações financeiras pessoais e comerciais, redes externas ou acesso a banco de dados, são apenas algumas das centenas de aplicações.
- *Software* de inteligência artificial - faz uso de algoritmos não numéricos para resolver problemas complexos que não sejam favoráveis à computação ou à análise direta. Atualmente a área de "*Artificial Inteligency - AI*" mais ativa é a dos "sistemas especialistas baseados em conhecimentos", porém outras áreas de aplicação para o *software* de AI são o reconhecimento de padrões (voz e imagem), jogos e demonstração de teoremas. Uma rede neural simula a estrutura dos processos cerebrais (a função do neurônio biológico) e pode levar a uma nova classe de *software* que consegue reconhecer padrões complexos e aprender com a "experiência" passada.

4 CICLO DE VIDA DO SOFTWARE

O ciclo de vida clássico[Lif80], [Pre97], [Som92] está demostrado na figura 1, ilustrando o seu paradigma. Às vezes chamado modelo cascata, o paradigma do ciclo de vida requer uma abordagem sistemática, seqüencial ao desenvolvimento do *software* que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção. Modelado em função do ciclo da engenharia convencional, o paradigma do ciclo de vida abrange as seguintes atividades:

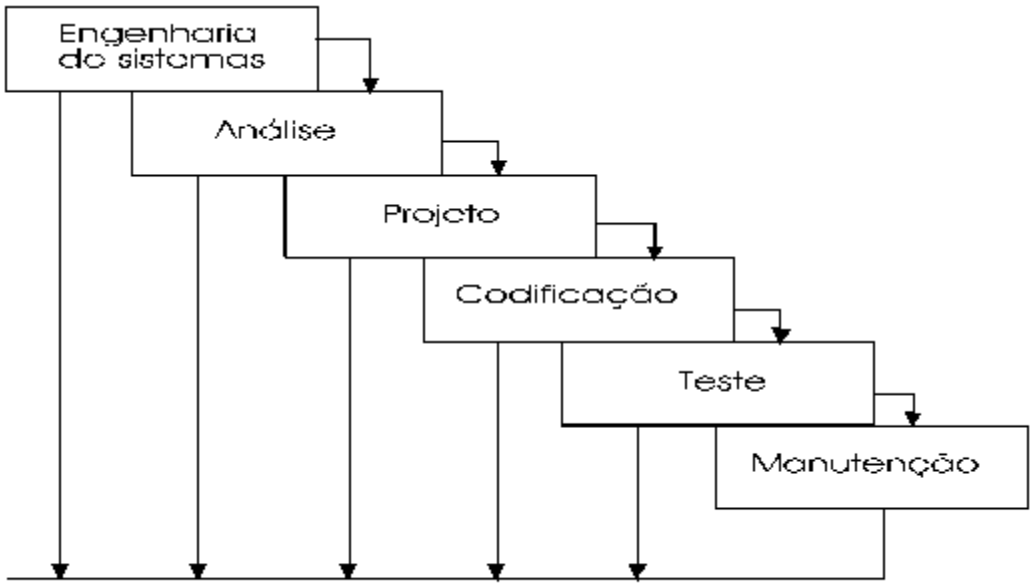


Figura 1 – O ciclo de vida clássico

4.1 Engenharia de Sistemas

Uma vez que o *software* sempre faz parte de um sistema mais amplo, o trabalho inicia-se com o estabelecimento dos requisitos para todos os elementos do sistema e prossegue com a atribuição de certo subconjunto desses requisitos ao *software*. Essa visão é essencial quando o *software* deve fazer interface com outros elementos, tais como hardware, pessoas e bancos de dados. A engenharia de sistemas envolve a coleta dos requisitos em nível do sistema, com uma pequena quantidade de projeto e análise de alto nível.

4.2 Análise

O processo de coleta dos requisitos é intensificado e concentrado especificamente no *software*. Para entender a natureza dos programas a serem construídos, o engenheiro (analista) de *software* deve compreender o domínio da informação para o *software*, bem como a função, desempenho e interface exigidos. Os requisitos tanto para o sistema como para o *software*, são documentados e revistos com o cliente.

4.3 Projeto

O projeto de *software* é de fato um processo de múltiplos passos que se concentra em quatro atributos distintos do programa: estrutura de dados; arquitetura de *software*; detalhes procedimentais e caracterização de interface. O processo de leitura do projeto traduz as exigências numa representação do *software* que pode ser avaliada quanto à qualidade antes que a codificação se inicie. Como os requisitos, o projeto é documentado e torna-se parte da configuração do *software*.

4.4 Codificação

O projeto deve ser traduzido numa forma legível por máquina. A etapa de codificação executa essa tarefa. Se o projeto for executado detalhadamente, a codificação pode ser executada mecanicamente.

4.5 Testes

Assim que o código for gerado, inicia-se a realização de testes de programa. O processo de realização de testes concentra-se nos aspectos lógicos internos do *software*, garantindo que todas as instruções tenham sido testadas.

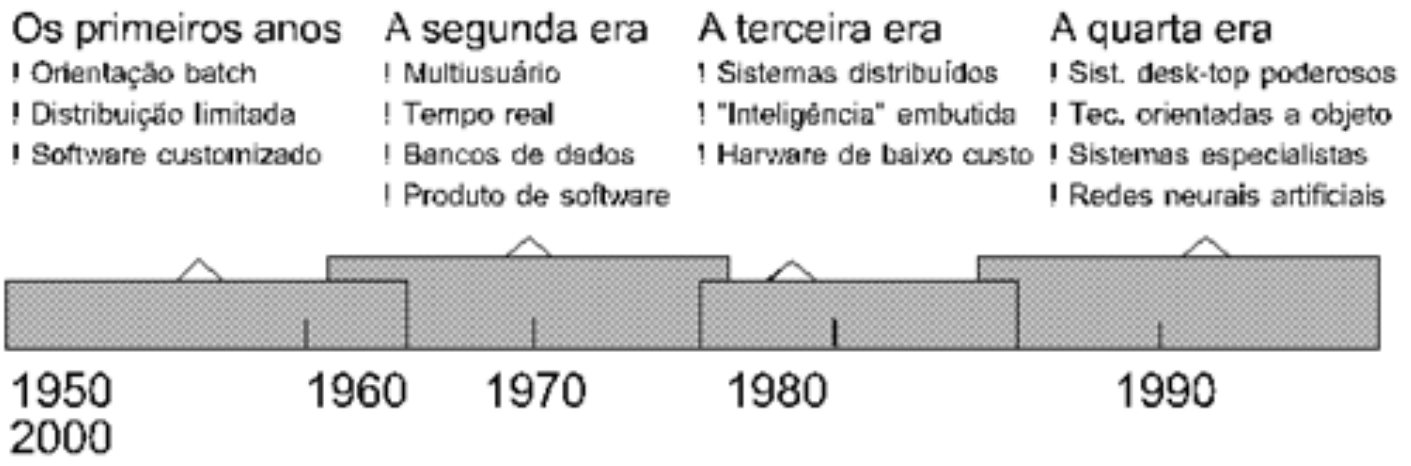
4.6 Manutenção

O *software* sofrerá mudanças depois que for entregue. Podem ocorrer mudanças porque erros foram encontrados ou devem ser adaptados, a fim de que sejam feitas mudanças em seu ambiente. A manutenção replica cada uma das etapas precedentes do ciclo de vida a um programa existente.

Resumindo, o ciclo de vida é o paradigma mais antigo e mais usado na engenharia de *software*. Ele continua sendo o modelo mais usado. Embora tenha fragilidades, ele é muito melhor que uma abordagem sem nenhum critério ao desenvolvimento de *software*.

5. O PAPEL EVOLUTIVO DO SOFTWARE

O contexto em que o *software*[Pre95] foi desenvolvido está estreitamente ligado a quase cinco décadas de evolução dos sistemas computadorizados. O melhor desempenho de *hardware*, o menor tamanho e o custo mais baixo, precipitaram o aparecimento de sistemas baseados em computadores mais sofisticados. Mudamo-nos dos processadores à válvula para os dispositivos microeletrônicos que são capazes de processar 200 milhões de instruções por segundo. A figura 2 a seguir, descreve a evolução do *software* dentro do contexto das áreas de aplicação de sistemas baseados em computador.



5.1 Os Primeiros Anos

No desenvolvimento de sistemas computadorizados, o *hardware* sofreu contínuas mudanças, enquanto o *software* era visto por muitos como uma reflexão posterior. A programação de computador era uma arte secundária para a qual havia poucos métodos sistemáticos. O desenvolvimento do *software* era feito, virtualmente, sem administração até que os prazos começassem a se esgotar e os custos a subir abruptamente. Durante esse período, era usada uma

orientação *batch* (em lote) para a maioria dos sistemas. Notáveis exceções foram os sistemas interativos, tais como o primeiro sistema da *American Airlines* e os sistemas de tempo real orientados à defesa, como o SAGE. Na maior parte, entretanto, o *hardware* dedicava-se à execução de um único programa que, por sua vez, dedicava-se a uma única aplicação específica. Também nos primeiros anos, o *hardware* de propósito geral tornara-se lugar-comum. O *software,* por outro lado, era projetado sob medida para cada aplicação e tinha uma distribuição relativamente limitada. O *software* dito "pacote", isto é, programas para serem vendidos a um ou mais clientes, estava em sua infância. A maior parte do *software* era desenvolvida e em última análise usada pela própria pessoa ou organização. Você escrevia-o, colocava-o em funcionamento e, se ele falhasse, era você quem o consertava. Por causa desse ambiente de *software* personalizado o projeto era processo implícito realizado no cérebro de alguém e a documentação muitas vezes não existia. Por justiça, entretanto, devemos reconhecer que alguns sistemas surpreendentes desenvolvidos naquela época permanecem em uso até hoje.

5.2 A Segunda Era

Os sistemas computadorizados estenderam-se de meados da década de 1960 até o final da década de 1970. A multiprogramação e os sistemas multiusuários introduziram novos conceitos de interação homem-máquina. As técnicas interativas abriram um novo mundo de aplicações e novos níveis de sofisticação de *software* e *hardware*. Sistemas de tempo real podiam coletar, analisar e transformar dados de múltiplas fontes, daí controlando processos e produzindo saída em milissegundos e não em minutos. Os avanços da armazenagem *on-line* levaram à primeira geração de sistemas de gerenciamento de banco de dados. Também foi caracterizada pelo uso do produto de *software* e pelo advento das "*software houses*". O *software* era desenvolvido para ampla distribuição num mercado interdisciplinar. Programas para *mainframes* e minicomputadores eram distribuídos para centenas e às vezes milhares de usuários. Muitos puseram-se a desenvolver pacotes de *software* e ganhar muito dinheiro. À medida em que o número de sistemas baseados em computador crescia, bibliotecas de *software* começaram a se expandir. Uma nuvem negra apareceu no horizonte. Todos esses programas, essas instruções, tinham que ser corrigidos quando eram detectadas falhas, alterados conforme as exigências do usuário se modificavam ou se adaptavam a um novo *hardware* que fosse comprado. Essas atividades foram chamadas coletivamente de "manutenção de *software"*. E, ainda pior, a natureza personalizada de muitos programas tornava-os virtualmente impossíveis de sofrerem manutenção. Uma "crise de *software*" agigantou-se no horizonte.

5.3 A Terceira Era

Os sistemas computadorizados começaram em meados da década de 1970 e continuam até hoje. Os sistemas distribuídos e múltiplos computadores, onde cada um, executando funções concorrentemente e comunicando-se um com o outro, aumentaram intensamente a complexidade dos sistemas baseados em computador. As redes globais, as comunicações digitais de largura de banda ("*bandawidth*") elevada e a crescente demanda de acesso "instantâneo" a dados exigem muito dos desenvolvedores de *software*. Também foi caracterizada pelo advento e o generalizado uso de microprocessadores, computadores pessoais e poderosas estações de trabalho "*workstations*" de mesa. O microprocessador gerou um amplo conjunto de produtos inteligentes. Do automóvel a fornos microondas, de robôs industriais a equipamentos para diagnóstico de soro sangüíneo. Em muitos casos, a tecnologia de *software* está sendo integrada a produtos por equipes técnicas que entendem de *hardware* mas que freqüentemente são principiantes em desenvolvimento de *software*. O computador pessoal foi o catalisador do crescimento de muitas empresas de *software*. Enquanto as empresas de *software* da segunda era vendiam centenas ou milhares de cópias de seus programas, as empresas da terceira era vendem dezenas e até mesmo centenas de milhares de cópias. O *hardware* de computador pessoal está se tornando rapidamente um produto primário, enquanto o *software* oferece a característica capaz de diferenciar. De fato, quando a taxa de crescimento das

vendas de computadores pessoais se estabilizou em meados da década de 1980, as vendas de *software* continuaram a crescer.

5.4 A Quarta Era

Esta era está apenas começando. As tecnologias orientadas a objetos, orientadas a documentos, estão ocupando o lugar das abordagens mais convencionais para o desenvolvimento de *software* em muitas áreas de aplicação. As técnicas de "quarta geração" para o desenvolvimento de *software* já estão mudando a maneira segundo a qual alguns segmentos da comunidade de *software* constroem programas de computador. Os sistemas especialistas e o *software* de inteligência artificial finalmente saíram do laboratório para a aplicação prática em problemas de amplo espectro do mundo real. O *software* de rede neural artificial abriu excitantes possibilidades para o reconhecimento de padrões e para capacidades de processamento de informações semelhantes às humanas.

6 CONCLUSÃO

Durante as três primeiras décadas da era do computador, o principal desafio era desenvolver um *hardware* que reduzisse o custo de processamento e armazenagem de dados. Ao longo da década de 1980, avanços na microeletrônica resultaram em maior poder de computação a um custo cada vez mais baixo. Hoje o problema é diferente. O principal desafio durante a década de 1990 é melhorar a qualidade e reduzir o custo de soluções baseadas em computador. Soluções que são implementadas com *software*. O poder de um computador *mainframe* da década de 1980 agora está à disposição sobre uma escrivaninha. As assombrosas capacidades de processamento e armazenagem do moderno hardware representam um grande potencial de computação. O *software* é o mecanismo que nos possibilita aproveitar e dar vazão a esse potencial.

Segundo Verzello[ver84], o *software* pode ser classificado em três tipos, denominando-os de "sistema", o qual conceitua-os como controlador e coordenador do *software;* de "linguagens", o qual diz que traduzem outros programas escritos em linguagens de programação e de "aplicação", que servem para resolver problemas comerciais ou prestar outros serviços de processamento de dados aos usuários.

Para Pressman[Pre95], que amplia esta classificação para 7 categorias, denominando-as de: "básico", sendo uma coleção de programas escritos para dar apoio a outros programas; "tempo real", onde monitora, analisa e controla eventos do mundo real; "comercial", onde segundo Pressman, é a maior área particular de *software;* "científico", o qual se caraterizada por algoritmos de processamento de números; "embutido", para controlar produtos e sistemas para os mercados industriais e de consumo; "computador pessoal", sendo os *softwares* para computadores pessoais; "inteligência artificial", fazendo uso de algoritmos não numéricos para resolver problemas complexos que não sejam favoráveis à computação ou à análise direta.

O próprio Pressman admite ser uma tarefa um tanto difícil desenvolver categorias genéricas para aplicações de *software*, opinião com a qual nós concordamos e vamos fazer algumas considerações.

Como podemos observar, a classificação de Pressman se baseia no uso da tecnologia e no seu tipo de utilização.

Se assumirmos que essa forma de classificação está dentro da nossa linha de pensamento, poderíamos também citar novas categorias como, por exemplo: *software* para rede de computadores, *software* de controle de tráfego aéreo ou esta nova categoria que está surgindo de *software* denominado "*software* robô", que são os sistemas desenvolvidos para navegar na rede mundial de computadores, a *Internet,* onde a sua principal atividade é sair vasculhando os

computadores de todo o mundo, procurando trabalhos desenvolvidos por pesquisadores ou profissionais para depois poder referenciá-los em uma busca qualquer. Por exemplo, o *site* da "A*lta Vista Digital*" ou "*Yhaoo*", que têm vários "*robosoft*" que saem pela rede "*Internet*" buscando assuntos para depois poder fazer referência em suas pesquisas.

Concluindo, será que a visão de Pressman "desvirtua" ou "amplia" o sentido de classificação ou categoria de evolução de *software*?

7 REFERÊNCIAS BIBLIOGRÁFICAS

[Li 80] LIFE cyde management : analysis (Invited Papers). Infotech State of the art, Series 8, Number 7, 1980.

[Pre95] PRESSMAN, Roger S. **Engenharia de software**. São Paulo : Makron, 1995.

[Pre97] PRESSMAN, Roger S. **Software engineering**: a practiotioner's approach. New York: Makron, 1997.

[Sho83] SHOOMAN, Martin L. **Software engineerings.** .singapore : McGraw-Hill, 1983.

[Som92] SOMERVILLE, Iam. **Software engineering**. 4. ed. Addison-Wesley, 1992.

[Ver84] VERZELLO, Robert J.; REUTTER III, John **Processamento de dados**. São Paulo : McGraw-Hill, 1984.

QUESTIONÁRIO

1 - Explique o que quer dizer com dois tipos de componentes.

2 - O que é a reusabilidade, explique esse conceito em SW.

3 - O que é "Classe" e "Objeto" ?

4 - Explique o que : Linguagem de maquina, alto nivel e não procedimentais?

5 - Qual a relação entre Software e o automóvel sugerida pelo autor?

6 - Explique as três classificações de SW por Verzello.

7 - Quais as categorias de sw por Pressman ? Explique cada um das 7.

8 - O que é e qual o conceito do RoboSoft ?

9 - Quais as principais etapas no Ciclo de Vida de SW ?

10 - Sintetize as 4 etapas ou eras evolutivas do SW.