Universidade Eduardo Mondlane

Faculdade de Ciências

Departamento de Matemática e Informática

Curso de Informática

Trabalho de Licenciatura

Migração de Aplicações de um ambiente Informix 4GL baseado em caracteres para um ambiente gráfico, utilizando a Four J's Business Development Language (BDL)

Aramuge, Esmeraldo Temóteo Gonçalves

Universidade Eduardo Mondlane

Faculdade de Ciências

Departamento de Matemática e Informática

Curso de Informática

Trabalho de Licenciatura

Migração de Aplicações de um ambiente Informix 4GL baseado em caracteres para um ambiente gráfico, utilizando a Four J's Business Development Language (BDL)

Supervisores:

dr.ª Marisa Balas

Eng.º José Grachane

Co-Supervisor:

dr. Carlos Cumbana

Autor:

Aramuge, Esmeraldo Temóteo Gonçalves

Maputo, Julho de 2005

AGRADECIMENTOS

A todos aqueles que me ajudaram de alguma forma no desenvolvimento deste trabalho vai o meu profundo agradecimento.

Aos meus Pais, Jacinto Aramuge e Maria José Gonçalves pela vida, cuidado, apoio moral e material que me deram e continuam dando, deixo a minha maior expressão de gratidão com amor único e insubstituível. "Quero-vos muito!" "Quero vosso colo!"

À minha madrasta, alias "minha mãe", Virgínia Paulo Ferreira de Aramuge, pela mamã que foi, por "cinco mais um" anos vivi o adorável cuidado seu, durante a minha escolaridade do 1° Ciclo, fica o meu obrigado com abraço e muitos beijinhos.

E a minha avó Adélia Gonçalves (*in memoriam*), pelo cuidado, carinho e ensinamentos de moral e civismo que deu-me, quando viva, no período que compreendeu o ensino primário do 1° grau – que Deus a tenha e salve a sua alma!

Agradeço a minha "namorada" Jorgina António (Menna), pela tolerância com minhas angústias, principalmente pelo amor e força demonstrados em cada momento ao longo do tempo da realização deste Trabalho de Licenciatura.

Em especial agradeço ao dr. Zeferino Saugene (MSc.), à dr.ª Carla Xavier, ao Sr. Jordão Machoco, ao dr. Elias Nhancale e a todos os colegas na EXI pela elevada colaboração que vieram dando no sentido de constituir este trabalho uma realidade e a minha pessoa um profissional em tecnologias de informação, Analista e Programador de Sistema de Informação. Muito obrigado por serem orientadores e incentivadores de todos os momentos, sempre prontos a me indicar o bom caminho.

À equipe da Biblioteca do DMI, na disponibilizarão de material utilizado neste trabalho e no decurso da carreira escolar, e, com toda consideração e respeito, ao corpo docente do DMI, que não poupa esforço a dar para a causa do conhecimento e da ciência para o desenvolvimento da UEM e de Moçambique.

Ao dr. Carlos Cumbana (MSc), ao Eng.° José Grachane, à dr.ª Marisa Balas (MSc), meus supervisores, vão os meus profundos agradecimentos pela atenção, empenho, paciência e tempo disponibilizado no intuito de tornar este trabalho um sucesso.

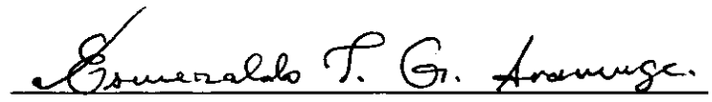
Agradeço ao Prezado Sr. Bonifácio Gruveta, ao Sr. Rogério Gaspar e sua esposa por terem contribuído para a minha deslocação a Maputo para ingressar no ensino superior.

DECLARAÇÃO DE HONRA

Declaro por minha honra, que este trabalho é resultado da minha investigação que não foi submetido para outro grau que não seja o de Licenciatura em Informática na Universidade Eduardo Mondlane.

Maputo, Julho de 2005

O Aluno



(Esmeraldo Temóteo Gonçalves Aramuge)

RESUMO

A evolução da informática, nas últimas décadas, revolucionou a maneira como o utilizador interage com o computador (interface do utilizador) de modo a aproveitar as maiores capacidades de processamento dos computadores e integrarem mais capacidades humanas. A adopção da *tecnologia Graphics User Interface* (GUI – interface gráfica) permitiu encontrar a solução para as interfaces baseadas em caracteres das aplicações Informix 4GL (I4GL), como é o caso das interfaces no Sistema de Informação para a Gestão de Finanças Públicas (Sis-Pública) da empresa EXI (Engenharia e Comercialização de Sistemas Informáticos) e noutros sistemas de empresas do mercado nacional ou internacional.

Este estudo, tem a proposta de fornecer informação necessária para disponibilizar uma solução que atenda a necessidade de migração de interface de aplicações, de um ambiente I4GL baseado em caracteres para um ambiente gráfico ("agradável"), utilizando a Four J's *Business Development Language* (BDL) adquirida para tal.

Em vista do referido nos parágrafos anteriores, aplicaram-se as tecnologias que transformam as interfaces baseadas em caracteres das aplicações I4GL para interfaces gráficas ou para Web, mantendo as aplicações originais inalteradas e a funcionalidade existente, preservando, deste modo, os investimentos e o legado das aplicações. A abordagem teórica fundamentou-se no estudo de conceitos sobre Evolução de Sistemas, Reengenharia de Sistemas de Informação, Modelação de Sistemas, Sistemas em ambiente Unix, Arquitetura Cliente/Servidor e Ferramentas de Migração (Freedom, Informix *Dynamic* 4gl e Four J's *Business Development Language* (BDL)).

Ao utilizar a Evolução de Sistemas aplicou-se a Migração de Interface que é decorrente da Reegenharia de Software, uma vez que os custos e os riscos de substituição do sistema são elevados, e a manutenção continua não converte as interfaces dos sistemas.

A implementação do modelo obtido aumentou o ciclo de vida do sistema, assegurou a "agradabilidade" da interface e a melhor razão custo benefício dos investimentos realizados, como também, traz benefícios para a EXI e seus clientes ao incorporar um *item* de qualidade (interface gráfica) às aplicações no Sis-Pública. E a avaliação desta implementação possibilitou a confirmação da aplicabilidade da ferramenta de migração (Four J's BDL) que foi escolhida entre as três mencionadas.

ÍNDICE

1. INTRODUÇÃO

> *"Não é o mais forte da espécie que sobrevive, nem o mais inteligente: é o que melhor se adapta às mudanças" (A origem das espécies, 1859, Charles Darwin).*

Nas últimas décadas a informática tem evoluído com velocidade sem paralelo na história. Novas tecnologias são lançadas uma após outra, e após a sua absorção pelo mercado, acabam surgindo novas necessidades que demandam os seus aprimoramentos ou substituições. As empresas que não conseguirem acompanhar estas constantes alterações de cenário com a rapidez que lhes é exigida, estão sujeitas a serem superadas pelas concorrentes mais ágeis.

Paralelamente a este desenvolvimento, as Interfaces de Utilizador têm evoluído para aproveitar as maiores capacidades de processamento dos computadores no sentido de integrarem mais capacidades humanas.

As primeiras interfaces baseadas em caracteres foram a solução óbvia num contexto onde a principal preocupação no desenvolvimento de aplicações era o de optimizar o tempo de computação e memória. O aparecimento das primeiras interfaces gráficas demonstrou uma maior preocupação com a qualidade das interacções Pessoa-Computador [Imai,1992].

Para ilustrar rapidamente como se chegou a este quadro será transcorrida brevemente, nos parágrafos a seguir, a história da informática em termos de apresentação da informação (Interfaces), dentro das últimas décadas.

No período compreendido entre os anos sessenta e meados dos anos setenta, poucas empresas tinham condições de possuir um computador devido a toda estrutura que era necessária para mantê-los, principalmente pelos altos custos envolvidos. As apresentações das informações baseavam-se em *ecrans* de caracteres exibidas em monitores monocromáticos, chamados de terminais não inteligentes, que não possuíam capacidade de processamento [Da Luz, Chaves e Nunes. 2001].

Vieram os anos oitenta, e com eles, o surgimento dos *Personal Computers* (PC). É nesta época que ocorreu a popularização dos micro-computadores, e em seguida, a proliferação das redes de computadores[Da Luz, Chaves e Nunes. 2001].

As opções de tecnológicas cresceram, todavia, os problemas que ocorriam por consequência dos sistemas de informação proprietários persistiam. A interface homem-máquina acabou tornando-se mais amigável após a adopção da tecnologia *Graphics User Interface* (GUI – interface gráfica). A partir deste momento os usuários de computadores começaram a ficar mais exigentes quanto à

interacção com os programas, e cada vez mais perderam o interesse de trabalhar com os *ecrans* de apresentação por caracteres [Da Luz, Chaves e Nunes. 2001].

Um dos problemas que as empresas enfrentavam era o elevado custo de substituição dos seus sistemas antigos, com interface baseado em caracteres, cuja funcionalidade havia sido optimizada ao longo dos anos, por sistemas GUI. E, para providenciar as aplicações em Informix 4GL com interfaces GUI, em 1995 foram criados, a *Business Development Language* (BDL) e o compilador da Four J´s.

A Four J´s proporciona um ambiente integrado de desenvolvimento (IDE) que recompila o código Informix 4GL para interoperar com um certo número de bases de dados, incluindo Informix, DB2, Oracle e MS SQL [URL-4]. Isto permite aos desenvolvedores estender a funcionalidade e conectividade das aplicações existentes, e construir novas aplicações que operam em ambientes mistos.

A Four J's, uma companhia fundada em 1995, ajuda outras empresas a adaptar as suas aplicações para a nova economia graças a inovadora e flexível linguagem de desenvolvimento de negócio (BDL), suportando integrações sem emendas com os mais recentes padrões tais como HTML, Java, WAP/WML e XML [URL-4].

E, hoje em dia, com a desmaterialização das empresas, onde o *focus* passou a estar centrado nos seus Sistemas de Informação e no valor intrínseco da Informação, qualquer processo de mudança substancial dos processos de negócio, bem como a migração da interface, não pode estar desassociado de um processo de Reengenharia de Sistemas de Informação, embora que nem sempre a Reengenharia de Sistemas de Informação (RSI) seja uma consequência da Reengenharia de Processos de Negócio.

A Reengenharia de Sistemas de Informação surge como uma nova necessidade decorrente da Reengenharia de Processos de Negócio (RPN), conceito que surgiu há 10 anos e foi implementado ao longo da década de 90 [URL-1].

A EXI – Engenharia e Comercialização de Sistemas Informáticos, é uma empresa detentora de experiência de prestação de serviços na área de Tecnologias de Informação e Comunicação, há mais de 10 anos, operando no mercado nacional. Está estruturada em 3 unidades produtivas: Software Aplicacional, Infra-estrutura e Software de Sistema, e, *Procurement* e Vendas. Tem um perfil virado para a concepção, construção e operação de sistemas de informação. Ao empregar tecnologias de informação mais adequadas, desenvolveu um sistema de informação para a gestão de finanças públicas, denominado Sis-Pública, usando a Linguagem Informix 4GL. Há adopção deste

sistema por parte da Universidade Eduardo Mondlane (UEM), Instituto Nacional do Cajú (INCAJU) e do Projecto de Desenvolvimento Empresarial (PODE), e a interface em uso é baseada em caracter.

Foi neste cenário, onde a cada momento surgem novas tecnologias, que se verificou a necessidade da EXI possuir mecanismos que lhe permitam garantir vantagens competitivas aos seus clientes em Moçambique.

A base dos mecanismos referidos é a implementação de uma Interface Gráfica, *Graphical User Interface* (GUI), "agradável", no Sis-Pública, que deve ser de baixo custo e ter capacidade de evolução sem que haja necessidade de muitos investimentos em treinamento ou infra-estrutura, pois muitas empresas, a semelhança da EXI, não podem (ou não querem) fazer novos investimentos toda a vez que houver uma novidade em termos de tecnologias. Em tempos atrás, entre 1997 a 2001, tiveram lugar actividades de desenvolvimento de trabalhos do género utilizando a Informix *Dynamic* 4gl, na sua versão 3.0, bem como a Four J's *Business Development Language*(BDL), nas suas versões 3.10.4e e 3.20.2c, que não satisfizeram as necessidades da EXI. Agora o trabalho insere-se na versão 3.53.1a do compilador da Four J's.

A proposta deste trabalho é a de fornecer informações necessárias para prover uma solução que atenda às necessidades de migração de aplicações I4GL, de um ambiente baseado em caracteres para um ambiente gráfico, utilizando a Four J's BDL. E esta migração é feita ao abrigo da Reengenharia de Sistemas de Informação.

Elaborou-se, para efeito de estudo, uma descrição/definição dos principais termos que serão utilizados durante os próximos capítulos e se encontra no anexo I. No anexo V apresentam-se um glossário e a definição de termos comuns.

O trabalho comporta a seguinte estrutura: Inicia com a redacção introdutória deste Trabalho de Licenciatura, no presente capítulo (primeiro capítulo). No segundo capítulo é apresentado o problema no qual este trabalho se baseia e a caracterização do mesmo. No terceiro capítulo são definidos os objectivos que este trabalho pretende alcançar. No quarto capítulo são apresentados os materiais e os métodos usados para a realização do presente Trabalho de Licenciatura. No quinto capítulo são expostas as teorias relacionadas as tecnologias necessárias para resolver os problemas apresentados, os conceitos e os aspectos comuns às comunicações Cliente-Servidor. Já no sexto capítulo, é apresentado um conjunto de algumas ferramentas de migração de interfaces que consubstanciam uma solução se utilizadas com as tecnologias referidas nos capítulos anteriores, e também faz-se a fundamentação da escolha de uma das ferramentas a empregar no capítulo

seguinte. No sétimo capítulo é desenvolvido um caso de estudo em que se implementa uma solução baseando-se no conteúdo deste trabalho. Para finalizar, no oitavo capítulo são apresentadas as conclusões e recomendações deste trabalho.

2. DEFINIÇÃO DO PROBLEMA

A interface baseada em caracteres, em uso no Sistema de Informação para a Gestão de Finanças Públicas (Sis-Pública), é um conceito considerado antiquado, embora que o "shell" do Sistema Operativo *Unix* (ambiente onde funciona o Sis-Pública em Informix 4GL) ofereça mecanismos para tornar a vida do administrador ou usuário do Sistema de Informação muito simples e produtiva, e seja ainda uma ferramenta de valor inestimável para operações em máquinas remotas.

Estas considerações, levam à identificação da necessidade de abandonar-se o ambiente de caracteres, sem *mouse*, que é executado num terminal, ou num servidor, para uma aplicação com total funcionalidade em ambiente gráfico que pode ser rodada num ambiente cliente-servidor ou na Web, sem importância de re-treinar os usuários, nem re-escrever o código das aplicações.

A preocupação de prolongar o ciclo de vida das aplicações já desenvolvidas em Informix 4GL, permitindo que estas aplicações possam ser convertidas de forma a utilizarem ferramentas suficientemente flexíveis para seguirem as tendências do negócio e da tecnologia, com tempo e custo reduzidos, é também evidente, atendendo que existem ferramentas, como a Four J's *Business Development Language* (BDL) da Four J's, apoiam a evolução dos sistemas com aplicações em questão.

2.1. CARACTERIZAÇÃO DO PROBLEMA

Como se pode verificar, na base do referido na definição do problema, é de encontrar na actualidade, pelos desenvolvedores de *software* de empresas de médio e grande porte, dificuldades na extracção de dados de bases de dados e na apresentação destes nas diversas plataformas existentes. A mesma preocupação pode ocorrer sobre a forma de troca de informação ou de comunicação que ainda possa ser com recurso à interface do utilizador baseado em caracteres.

No Sis-Pública, a dificuldade encontrada é a de prover uma interface homem-computador que apresenta de uma maneira amigável a informação que está contida nas bases de dados dos servidores e até nas estações de trabalhos, com recurso ao uso do *mouse*, a quando da manipulação dos dados nas aplicações I4GL. Em várias empresas e/ou instituições a informação é encontrada, mediante consultas efectuadas através de interfaces de utilizador baseada em caracteres, não utilizando as facilidades oferecidas pelas interfaces gráficas.

Tendo em vista a dificuldade citada, é necessário encontrar a solução para a interface baseada em caracteres, das aplicações I4GL, para posterior implementação no Sis-Pública, de modo a que esta possa prover uma interface gráfica, "amigável" e "agradável".

Na sequência do trabalho será feita uma abordagem sobre algumas tecnologias recentes que serão empregues para o desenvolvimento de uma solução para sanear a dificuldade (necessidade) até agora apresentada.

3. OBJECTIVOS

3.1 Objectivo Geral

Migrar a aplicação do Sistema de Informação para a Gestão de Finanças Públicas, Sis-Pública, de um ambiente Informix 4GL baseado em caracteres para um ambiente gráfico, utilizando a Four J's *Business Development Language* (BDL), mantendo a funcionalidade existente.

3.2 Objectivos Específicos

- Identificar os constrangimentos ergonométricos existentes no Sis-Pública actual;
- Desenhar um modelo de Conversão de Aplicações I4GL utilizando a ferramenta da Four J's, *Business Development Language* (BDL);
- Usar o modelo de conversão para implementar a migração no Sis-Pública;
- Avaliar a implementação.

4. MATERIAIS E MÉTODOS

Para a realização do presente Trabalho de Licenciatura aplicou-se uma metodologia faseada consubstanciada em dois grandes pilares – *Reverse* e *Forward Engineering*, entendendo-se com *Reverse Engineering* o processo de aprendizagem e re-documentação de alto nível de abstracção do sistema legado; e por *Forward Engineering* como um conjunto de actividades de redesenvolvimento e implementação dos novos requisitos [URL-1].

As actividades realizadas consistiram em:

- Recolha de dados, baseando-se nas seguintes técnicas:

  – Entrevistas não estruturadas, a utilizadores do Sis-Pública e funcionários do Departamento de *Software* da EXI, com vista a colher dados acerca da interface – opinião sobre a interface baseada em caracteres e recomendações para o projecto de uma interface gráfica;

  – Consultas bibliográficas, pesquisa de informação detalhada sobre as ferramentas de migração de Aplicações I4GL, bem como a linguagem I4GL e Four J's DBL;

  – Observação participativa de um procedimento de migração com técnicos especialistas de BDL.

- Concepção do modelo para Implementação executando:

  – *Download* e teste da ferramenta, culminando com aprendizagem da mesma;

  – Análise comparativa e qualitativa dos dados recolhidos e desenho do modelo de implementação da migração.

- Implementação e Teste

  – Fazendo o uso do modelo, deu-se lugar a implementação do mesmo, que consistiu na conversão automática de código em I4GL para ambiente a que se propõe no trabalho de licenciatura, alteração e adaptação dos formulários;

  - Avaliação dos resultados da implementação, com base no produto de todo o processo desenvolvido, relevando a importância dos testes, fazendo de parâmetro a funcionalidade equivalente da aplicação original.

Material usado para a implementação do trabalho de licenciatura, foi:

- Servidor de Base de Dados com Sistema Operativo *Unix* e a aplicação do Sis-Pública;

- Servidor de Base de Dados com Sistema Operativo *Windows*, para a aplicação do Sis-Pública convertida;
- Base de Dados *Informix Dynamic Server Workgroup Edition* 9.4, versão para *Windows*;
- Base de Dados *Informix Dynamic Server Workgroup Edition* 9.4, versão para *Unix*;
- Compilador C/C++ ;
- Ferramentas Four J's BDL (compilador e *runtime*), versão 3.53.1c.

Material usado para a documentação do trabalho, foi:

- Computador pessoal *Pentium* IV, 1.60GHz;
- Sistema Operativo *Microsoft Windows 2000*;
- *Microsoft Office 2000* (*Word, Excel e Power Point*);
- Impressora *HP DeskJet 640C*;
- Fotocopiadora.

5. FUNDAMENTAÇÃO TEÓRICA

Algumas das grandes empresas que utilizam a plataforma *mainframe* ou Cliente/Servidor estão aplicando as tecnologias que transformam a interface caracter das aplicações corporativas em interfaces gráficas ou para a *Web*, mantendo as aplicações originais inalteradas, preservando assim, os investimentos e o legado das aplicações.

Nesta parte do trabalho, são apresentados com mais detalhes os padrões básicos relativos as tecnologias que iremos utilizar no desenvolvimento de uma solução que permita prover uma interface de utilizador gráfica para a apresentação de informação de bases de dados instaladas em servidores de dados, com aplicações em Informix 4GL.

Também são apresentados os conceitos que são aplicados no desenvolvimento da solução referida/requerida.

5.1 Evolução de Sistemas

Actualmente, é comum encontrar sistemas de processamento de dados que estão em funcionamento há mais de 30 anos. Este facto ajuda-nos a perceber que o tempo de vida de um sistema é um aspecto importante e que o mesmo deve estar preparado para sofrer alterações durante o seu ciclo de vida.

O processo de implementar um conjunto de alterações que o sistema pode sofrer depois deste ser entregue aos utilizadores finais é chamado de evolução de sistemas[Renaissance, 1998b]. Esta é considerada uma das actividades fundamentais para qualquer processo de modelação de uma aplicação e consome cerca de 60% dos custos de um sistema [Renaissance, 1998b].

Em muitos casos, o conhecimento do negócio, suas regras e procedimentos estão embebidos nos sistemas antigos, tornando-os críticos para o funcionamento da organização e a sua troca por um sistema novo poderá resultar em perda de conhecimento crítico para o negócio. Este risco pode ser evitado utilizando a evolução ou mesmo a reengenharia do sistema e não a sua troca.

A capacidade de evolução de um sistema é determinada pelas partes que o constituem (fig. 5.1).

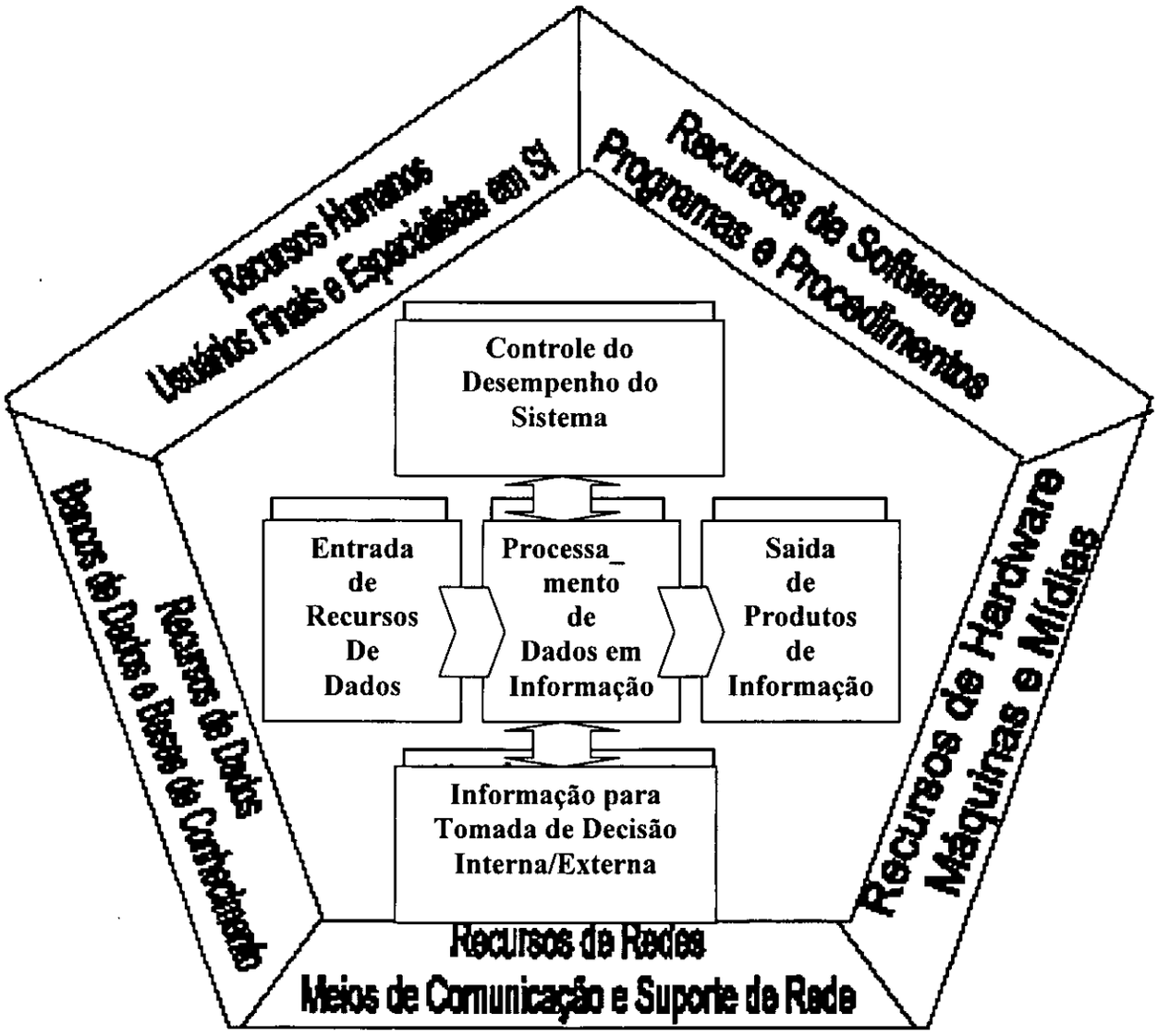


Fig. 5.1: Composição de um sistema em pleno funcionamento [TONORIO, 2004].

A evolução é um estado natural e necessário no ciclo de vida de um sistema. A fig. 5.2 ilustra os elementos que compõem o ambiente de um sistema em evolução e as suas interdependências.

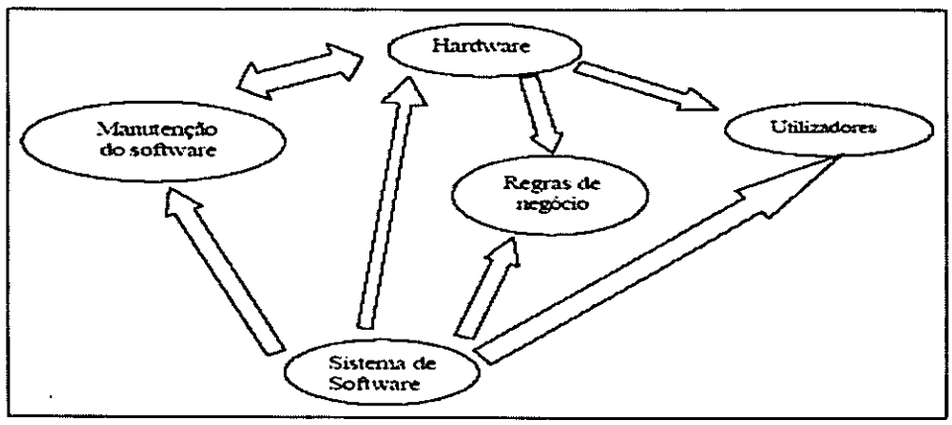


Fig. 5.2: Guias de evolução de um sistema e suas interdependências [Razak, 2001].

Os sistemas evolutivos devem ser desenvolvidos de acordo com processos bem definidos e baseados em princípios de engenharia, tais como documentação actualizada, flexibilidade em incorporar alterações, padronizações e possibilidade de usar componentes de outros sistemas.

Uma melhoria planeada do produto origina a necessidade de mudanças, isto é, novos requisitos deverão ser detectados durante a fase da sua recolha ou deverão ser extraídos dos objectivos da organização. Para que um processo de recolha de requisitos seja efectivo, necessita do envolvimento dos utilizadores do futuro sistema e da utilização de protótipos.

Os protótipos provaram ser uma técnica efectiva onde os utilizadores estão envolvidos desde o desenvolvimento até a validação dos seus requisitos, permitido-os destinguir o que eles pensam que necessitam e o que eles realmente necessitam. A falta de utilização desta técnica, nos sistemas legados (LS-*Legacy Systems*), pode ter sido a principal causa dos seus falhanços [Renaissance, 1998b].

Existem três grandes aproximações aos sistemas evolutivos que são [Renaissance, 1998b]:

- **Manutenção contínua**. Esta aproximação envolve uma manutenção contínua e consistente do sistema de acordo com o modelo tradicional de desenvolvimento;
- **Substituição de sistema**. O sistema existente é trocado por um novo. Esta aproximação pode ser necessária quando a reengenharia não é tecnicamente viável, ou por razões político-organizacionais, ou mudanças radicais nas regras do negócio;
- **Reengenharia de *software*** . É a transformação sistemática de um sistema existente em um novo sistema para melhorar a qualidade de operação, a capacidade do sistema, a funcionalidade, performance ou a evolução a baixo custo ou risco para o consumidor. Esta definição enfatiza que melhorias no sistema podem ser executadas com maior retorno do investimento do que com novos desenvolvimentos. Esta aproximação é geralmente aconselhada quando os novos desenvolvimentos acarretam riscos ou quando se pretende baixar os custos, tal é o contexto em que se insere o Sis-Pública.

5.1.1 Reengenharia de Software

A reengenharia de *software* é uma aproximação que somente é usada quando os custos e os riscos de substituição do sistema são elevados, é esta a razão de abordagem desenvolvida exclusivamente desta aproximação aos sistemas evolutivos, e porque com relação a migração de interface, esta embute como sendo uma das estratégias de reengenharia.

A reengenharia de *Software* é composta por quatro estratégias [Lancaster, 1998]: Migração da interface (" Re-vamping"), Reestruturação do sistema ("Restructuring"), Migração da arquitectura do sistema ("Rearchitecturing"), e Redesenho e reutilização de alguns componentes do sistema ("Redesign with Reuse").

5.1.1.1 Migração da Interface

É a estratégia que envolve a actualização ou substituição da interface do utilizador com o sistema. Ela é aplicável em organizações que pretendam migrar os interfaces dos seus sistemas para ambientes gráficos, sem alterar o código fonte existente. Esta migração é usada para melhorar a interface do sistema. É também usada por organizações que pretendem mover a interface baseado em caracteres, das suas aplicações que somente funcionam em terminais, para o ambiente gráfico dos computadores pessoais (*PC's*). Ela consiste em [Lancaster, 1998]:

1- A análise da interface do sistema e remoção de formulários redundantes;

2- O Redesenho do interface;

3- A Identificação e substituição de alguns componentes da interface;

4- A Interpretação automatizada da interface baseada em caracteres, de forma a produzir a interface gráfica.

A fig. 5.3 ilustra uma interface antiga, baseada em caracteres e as diversas possibilidades da sua evolução(migração).

Se o LS já possui a sua interface em modelo de camadas, a migração pode ser executada com base na opção 1 (alterando a interface, mas mantendo-a baseada em caracteres) ou a opção 2 (migrando-a para interface gráfica), ou seja, substituindo um ou mais componentes identificados.

Muitas vezes, é possível migrar a interface sem alterar a codificação existente. Isto é possível através do uso de *"Middleware"* (software que gere a interacção entre aplicações dispersas ao longo de plataformas heterogéneas [Lancaster,1998]), entre o LS e o novo *software* (Opção 4). Desta maneira, novas interfaces podem ser criadas independentemente das existentes e ainda, ambas poderão funcionar correctamente. Esta opção é geralmente usada quando o investimento em hardware é elevado e deverá ser escalonado, mantendo assim, alguns terminais não inteligentes em funcionamento como a antiga interface.

Geralmente nos LS, o código da interface com o utilizador (UI) é integrado com o restante código do sistema, o que torna difícil a sua alteração ou o seu redesenho. A solução para este caso estará na opção 3, ou seja, no desenvolvimento de novos componentes para o sistema.

Os componentes que se destacam quando se pretende migrar a interface de um sistema são [Lancaster, 1998]:

A *Interface com o utilizador* (GUI – interface gráfico, UIBC – interface baseado em caracteres), que é responsável pela representação externa do sistema;

O *Controlo de diálogo* (DC), componente que define a estrutura do diálogo entre o utilizador e o sistema;

O *Modelo de interface aplicacional* (IP), representação da aplicação sob ponto de vista aplicacional.

A migração da interface pode ter um impacto na actualização do hardware existente, como aquisição de terminais que suportem gráficos e de novos dispositivos de entrada e saída de informação ("Input/Output").

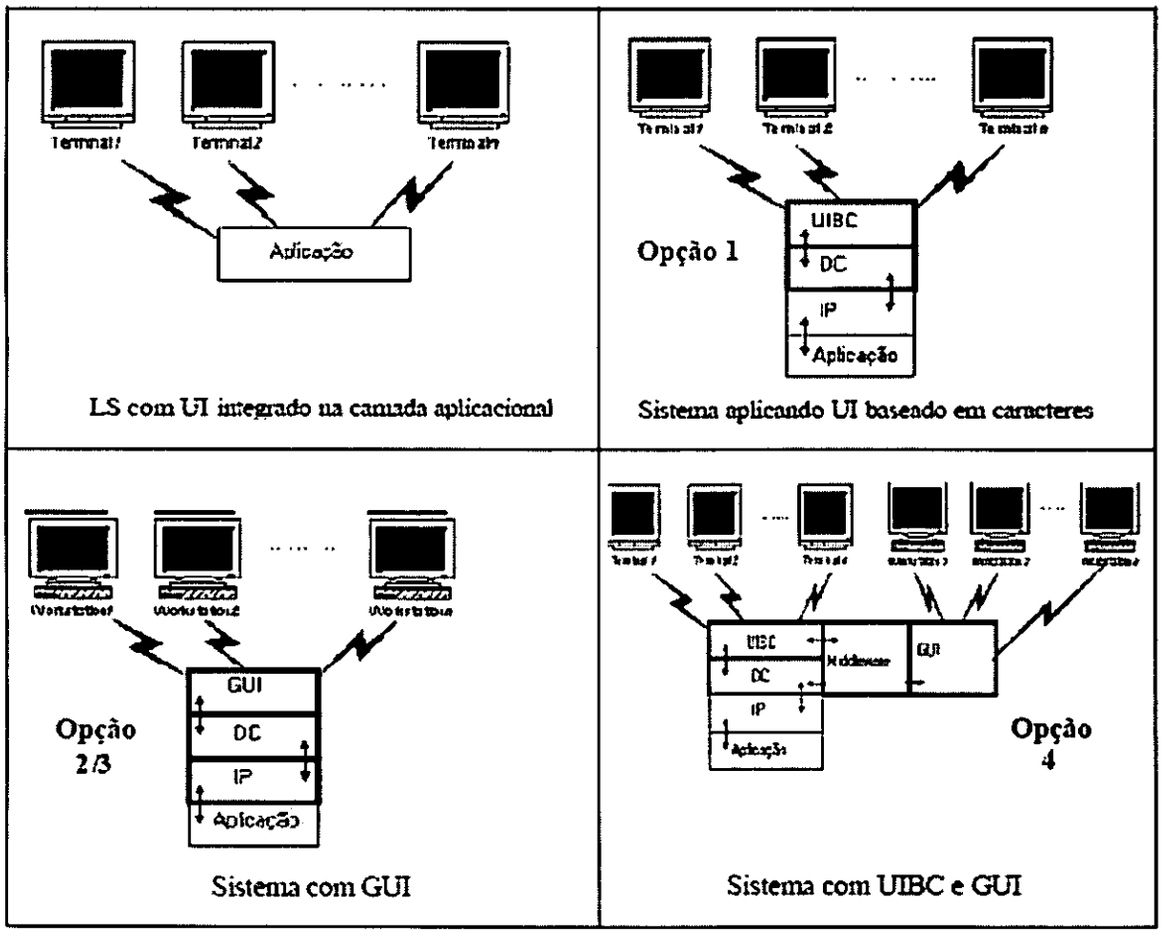


Fig. 5.3: LS e suas possibilidades de evolução da interface do sistema [Lancater,1998].

Segundo a Universidade de Lancaster [Lancaster, 1998], a migração da interface dá ao utilizador a impressão de ser uma nova aplicação, a sua implementação é mais fácil e não acarreta custos elevados, afecta minimamente o funcionamento da aplicação, e o LS com UI integrado na camada aplicacional (Sistema aplicando UI baseado em caracteres) facilita a manutenção da interface. Pode

ainda ser considerado o primeiro passo para a migração da arquitectura do sistema e cria bases para uma substituição incremental do sistema antigo.

Devido a estas qualidades, a migração da interface é mais usada em projectos de baixo custo e de curto prazo.

É de constatar que, na realidade, não existe uma nova aplicação, a codificação antiga continua por baixo da interface. Há aumento do tamanho da aplicação, devido a introdução da nova codificação (*middleware* e novo *interface*). A migração do interface nem sempre aumenta a capacidade de evolução do sistema e pode introduzir erros.

O risco principal deste sistema é que embora ele pareça ter uma imagem nova, continua a ser um LS. Isto é, os problemas de funcionalidade, se existirem continuarão a existir.

Outros riscos a ter em conta são os relacionados com a identificação, actualização e a possível remoção dos componentes escondidos dentro do sistema. Esta tarefa pode ser complexa e consumir bastante tempo. Por outro lado, usando "*middleware*" estes riscos podem ser reduzidos mantendo a interface antiga durante o processo de reengenharia e no sistema final.

5.1.1.2 Reestruturação do Sistema.

Esta estratégia é conhecida como a inversa da migração, ou seja, é a transformação da estrutura interna do sistema sem alterar a sua interface. Segundo Renaissance [Renaissance, 1998a], a reestruturação do sistema pode ser definida como uma transformação a partir de uma representação para a outra num mesmo nível de abstracção, enquanto se preserva o objectivo externo do sistema. Esta definição sugere implicitamente que a reestruturação pode ser feita em diferentes níveis de abstracção (requisitos, desenho e codificação), desde que se faça no mesmo nível e seja transparente para o utilizador.

A reestruturação do sistema envolve as seguintes etapas [Renaissance, 1998b]:

- Identificação e agrupamento do código que necessita de ser reorganizado;
- Remoção da redundância, melhoria na estrutura da codificação e no controlo de fluxo de dados;
- Reengenharia dos dados do sistema pela alteração da sua forma, por exemplo, move os dados de um gestor de base de dados hierárquico (DBMS - hierárquico) para um relacional (DBMS - relacional), ou para uma orientado a objectos (OODMS – orientado a objectos).

A estruturação do *Software* é uma colecção de factos que os engenheiros de *software* usam para desenvolver modelos mentais quando desenham ou tentam perceber o funcionamento do *software*

de um dado sistema. Estes factos incluem componentes do *software* como subsistemas; procedimentos e interfaces internos; dependências entre os componentes como restrições, forma de acesso aos dados e de acesso as funções e atributos específicos com tipo de componente e a localização do código fonte [Renaissance, 1998b].

A figura 5.4 ilustra como a estrutura interna de um LS poderá ser estruturada. É de referenciar que tanto o interface externo como a lógica aplicacional não sofrem qualquer alteração [Lancaster, 1998], isto é, a reestruturação não muda o sentido do programa ou a ordem da execução das instruções; mas muda a forma de como a lógica do programa foi escrita de modo a facilitar a sua compreensão e manutenção.

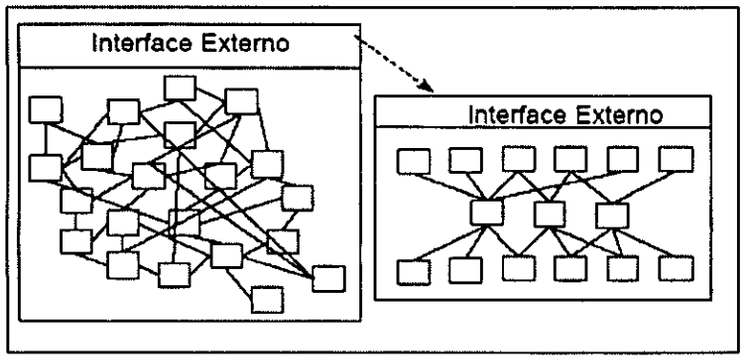


Fig.5.4: Reestruturação do LS [Lancaster,1998].

A reestruturação inclui o ordenamento de blocos de código de tal forma que secções logicamente relacionadas fiquem fisicamente próximas uma das outras, reduzindo os saltos no controlo do fluxo das operações e ainda envolve a replicação de pequenos fragmentos de código que melhoram a reestruturação do código [Lancaster,1998]. O resultado de uma reestruturação pode ser medido comparando a complexidade do sistema antes e depois da reestruturação.

Para que uma reestruturação seja efectuada, é necessário que tanto o sistema como a sua codificação estejam bem documentados ou que possam ser automaticamente analisados e transformados em processos automáticos. As principais vantagens desta estratégia são tornar o sistema mais eficiente, de fácil escalabilidade, extensível e seguro [Renaissance, 1998b].

A Universidade de Lancaster [Lancaster, 1998] ilustra que a reestruturação do sistema facilita a percepção do sistema, reduz os custos de manutenção, melhora a escalabilidade, a extensibilidade, a portabilidade e a segurança do sistema, torna o sistema evolutivo e actualiza a documentação sobre o sistema.

O que pode tirar certo mérito a esta estratégia está no facto de necessitar de pessoal com conhecimentos profundos sobre o LS, existência de poucas ferramentas de suporte. A possibilidade

de introdução de erros ao funcionamento do LS, a não introdução de nova funcionalidade ao sistema e a existência de código fonte não comentado ou de difícil percepção, constituem factos que também tiram méritos à reestruturação do sistema.

5.1.1.3 Migração de Arquitectura

A estratégia de migração envolve a mudança da arquitectura actual do sistema e afecta tanto o seu *software* como o seu hardware. Uma das estratégias da reengenharia, que é muito usada nas organizações, é a mudança de arquitectura, isto é, devido à limitação da flexibilidade da arquitectura centralizada, que consequentemente acarreta custos elevados de manutenção, as organizações, que possuem os seus sistemas sob esta, preferem migrá-los para uma arquitectura com maior flexibilidade, escalabilidade, suporte e partilha de recursos [Lancaster, 1998].

A mudança da arquitectura envolve [Lancaster, 1998]:

1- Análise da funcionalidade do sistema;

2- Identificação da funcionalidade que pode ser distribuída no servidor;

3- Mapeamento da funcionalidade através da estrutura do sistema;

4- Reorganização do sistema, criação do software cliente;

5- Mudança da arquitectura do código fonte (se necessário);

6- Produção de um documento actualizado do novo sistema.

A mudança de arquitectura é normalmente usada quando se pretende mover o sistema actual para uma nova arquitectura de hardware ou de software.

A RenaissanceWeb [Renaissance, 1998b] sugere que o processo de transformação da arquitectura do software de um sistema tem quatro passos:

1- *Recuperação do desenho (" design recovery").* Consiste em documentar um LS, identificando todos os seus pormenores;

2- *Modelação da aplicação ("application modelling").* Com base nos mesmos requisitos do LS, deve-se modelar o novo sistema;

3- *Mapeamento dos objectos (" object mapping").* Neste passo, efectua-se o mapeamento do desenho obtido no primeiro passo sobre o modelo criado no segundo passo;

4- Adaptação do código fonte ("Source-code adaptation"). Por último, adapta-se o código fonte de forma a implementar-se o passo anterior.

A figura 5.5 ilustra a migração de um sistema sob uma arquitectura centralizada para uma arquitectura distribuída.

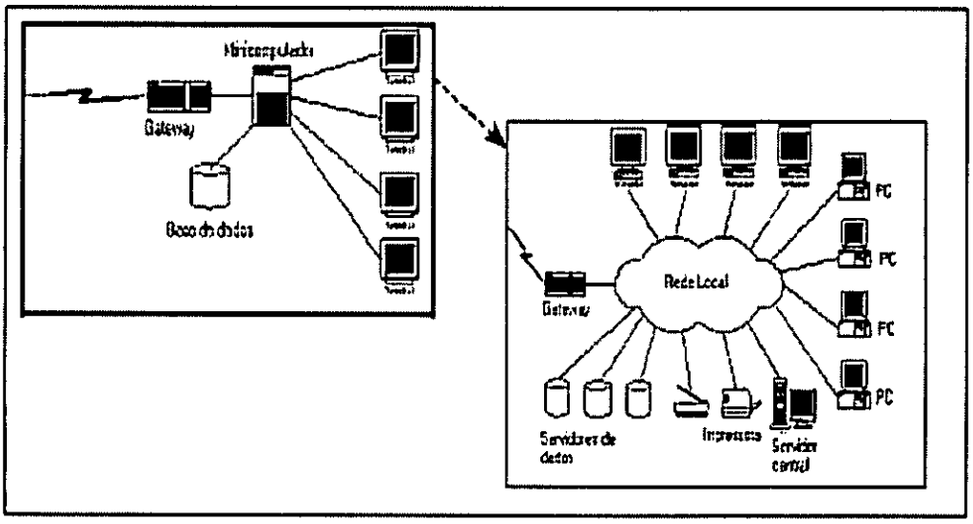


Fig. 5.5: Mudança de arquitectura de um LS [Lancaster, 1998].

Segundo a Universidade de Lancaster [Lancaster, 1998], esta estratégia possui todos méritos de uma reestruturação, e ainda é mais extensiva ao fazer o uso de tecnologias e arquitecturas modernas e poderosas, sem deixar de considerar que o sistema torna-se mais evolutivo.

Geralmente os LS operam com plataformas de hardware não padronizado ou ultrapassado.

Com a mudança da arquitectura, o novo sistema tornar-se-á mais flexível e sobre novas plataformas padrão e ainda responderá melhor às necessidades dos utilizadores.

A mudança de arquitectura tem a desvantagem de necessitar de grandes investimentos, necessidade de pessoal treinado.

Comparando com as duas estratégias anteriores, a mudança de arquitectura envolve tarefas que acarretam custos e com elevado grau de complexidade. Estas tarefas também envolvem grandes investimentos em hardware e contratação ou formação de pessoal familiarizado com a nova arquitectura.

5.1.1.4 Redesenho e Reutilização de Alguns Componentes do Sistema

Nesta estratégia, o sistema sofre uma transformação através de um novo desenvolvimento reutilizando alguns dos seus componentes. Na reengenharia do software, o redesenho e a reutilização de alguns componentes do LS (*"redesign and re-use"),* proporcionam a capacidade de manter o conhecimento embebido no LS e reduzir uma quantidade significativa de trabalho, uma vez que certos componentes poderão ser reutilizados no novo sistema (fig. 5.6) [Lancaster, 1998].

O *"Wrapping"* é o outro termo usado, na literatura, para identificar esta estratégia [Lancaster, 1998]. Ela adiciona uma interface fácil de perceber e ainda pode ser executada através de várias técnicas que tornam o código invisível para o utilizador ou para outros desenhadores.

As técnicas mais modernas e poderosas usam o paradigma orientado a objectos. Desta forma, o código antigo pode ser embebido num sistema moderno sob a arquitectura cliente/servidor de três camadas.

Esta estratégia envolve a identificação dos subsistemas críticos do LS e os seus encapsulamentos para a fase de redesenho do sistema, ou seja [Lancaster, 1998]:

- Identificação de subsistemas aproveitáveis para o novo sistema;
- Redesenho e definição do interface externo desses subsistemas (encapsulamento);
- Implementação dos subsistemas de modo a que estes sejam compatíveis com o novo sistema.

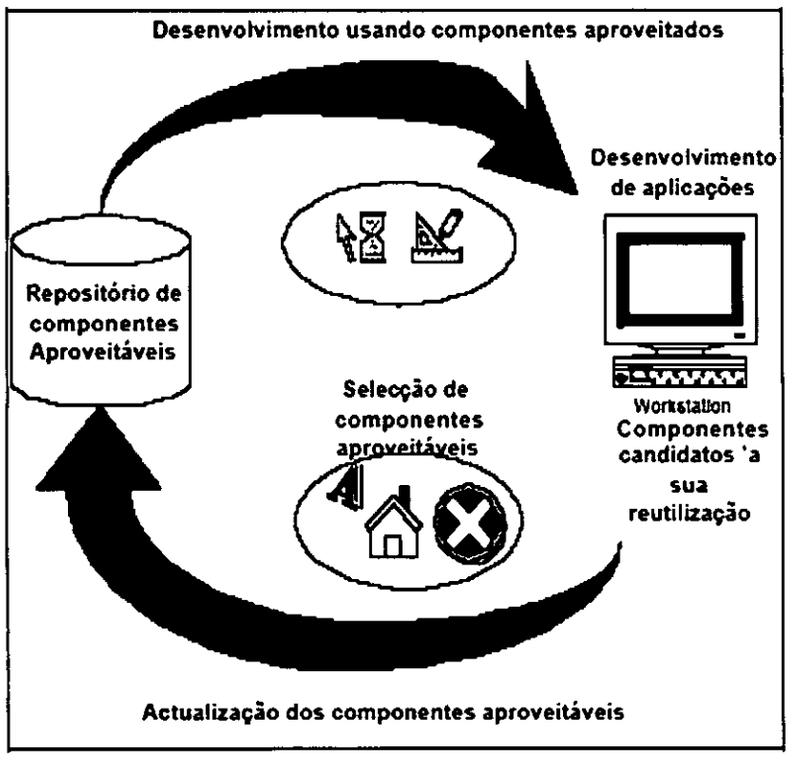


Fig. 5.6: Processo de desenvolvimento usando componentes de outros sistemas [Razak, 2001].

Segundo a Universidade de Lancaster [Lancaster, 1998], esta estratégia possui vantagens no concernente à reutilização de componentes existentes que minimiza as fases de desenho e de implementação, bem como os componentes existentes que podem ser usados em outros sistemas ou mesmo vendidos, e ainda o uso correcto dos componentes que pode aumentar o rendimento do novo sistema.

Por outro lado, a dificuldade na identificação de componentes aproveitáveis, constitui entrave nesta estratégia, bem como o código antigo que continua no sistema, e que pode originar certos conflitos (erros do sistema), para além do risco que se corre de retirar componentes que funcionam correctamente.

O que torna esta estratégia pouco aplicável é a tomada de decisão sobre os componentes a serem aproveitados, pois esta decisão envolve a análise da estabilidade do código, do suporte de funcionalidade, da documentação, das linguagens usadas na sua implementação e dos padrões existentes [Lancaster, 1998].

Secundando Razak (2001), a combinação das estratégias de evolução pode ser a solução ideal para a evolução de um LS, se atender-se o facto de que diferentes estratégias podem ser aplicadas em diferentes partes do sistema. No caso do Sistema de Informação para a Gestão de Finanças Públicas (Sis-Pública) da EXI, este responde correctamente às necessidades funcionais dos utilizadores. No entanto, seu único inconveniente, a interface de utilizador é baseada em caracteres. Para este caso, a evolução destes sistemas pode passar pela reengenharia da interface e a manutenção contínua da lógica aplicacional.

Referiu-se acima que, em muitos casos, o conhecimento do negócio, suas regras e procedimentos estão embebidos nos sistemas antigos, tornando-os críticos para o funcionamento da organização e a sua troca por um sistema novo poderá resultar em perda de conhecimento crítico para o negócio. Este risco pode ser evitado utilizando a evolução ou mesmo a reengenharia do sistema e não a sua troca. Sendo assim, a seguir debruçar-se-á sobre a reengenharia de sistemas de informação, solução optada por empresas em busca de uma maior optimização dos recursos informáticos, maior portabilidade e um rácio de manutenção custo/benefício mais atraente.

5.2 Reengenharia de Sistemas de Informação/Reengenharia de Processo de Negócio

5.2.1 Reengenharia de Processo de Negócio

O termo "reengenharia" faz lembrar um outro tipo de exercício, em voga na década de 80, que procurava essencialmente reduzir a dimensão da empresa (sobretudo em recursos humanos) através de uma reorientação de objectivos e funções e de uma reestruturação organizativa [URL-4]. No fundo, estava em causa concentrar a actividade da empresa, eventualmente abandonando algumas áreas, de forma a diminuir custos. A pergunta fundamental a responder era: "como fazer com menos custos o que sabemos fazer melhor?" [URL-4]

A reengenharia de processos de que está-se a falar não visa o mesmo tipo de soluções. Admite-se que a empresa faz eficientemente aquilo que faz. A pergunta a responder é "como se pode fazer

melhor aquilo que se faz no sentido de melhorar as posições no mercado?". Logo à partida, deve-se mesmo questionar se está-se a fazer o que se devia, isto é, se os produtos (bens ou serviços) estão adequadamente direccionados ao mercado, actual ou futuro (exercício algo especulativo).

Os processos de que se estão a falar não são apenas produtivo-tecnológicos, mas todos os conjuntos de tarefas logicamente relacionadas que são executadas para alcançar um determinado objectivo empresarial (alguns exemplos: desenvolvimento de um novo produto, requisição de materiais a um fornecedor, criação de um plano de *marketing*, negociação de um contrato de vendas) [URL-4]. Frequentemente, os processos tecnológicos são os que acabam por sofrer menos alterações no decurso de um *business process reengineering* (BPR - Reengenharia de Processos de Negócio) [URL-4].

"Um plano global de inovação pode ser desenhado a partir de um *Business Process Reengineering* (BPR): análise e projecto dos fluxos de trabalho e processos dentro da empresa e entre esta e o exterior" (Davenport e Short, 1990).

Existem duas opções possíveis em relação a um BPR: A metodologia incremental e a metodologia radical.

A abordagem incremental assemelha-se a um trabalho geral de estratégia (neste caso, utiliza-se mais frequentemente o termo BPI, em vez de BPR (Davenport, 1994). Enquanto que numa abordagem radical, tenta-se definir os processos da empresa a partir do zero, como se ela não existisse e tivéssemos de a construir de novo (Davenport, 1994).

Uma consequência relativamente óbvia do BPR é a utilização extensiva dos modernos sistemas de informação (IT- *information technologies*). No entanto, a melhoria dos processos não ocorre necessariamente apenas no modo de gerir a informação e a comunicação.

É importante notar que a reengenharia de processos pode conduzir a um novo projecto totalmente revolucionário na organização da empresa e suas operações, mas a sua implementação não necessita de ser radical. Possivelmente, isso implicaria custos incomportáveis. Daí que uma implementação faseada e planeada de acordo com as realidades e interesses da empresa constitua muitas vezes a estratégia mais adequada, a usada pela EXI.

Além disso, a abordagem teórica mais radical (a partir do zero absoluto) é mesmo de questionar actualmente, face à situação real da maioria das empresas e aos custos do *redesigning* (que é uma tarefa de conhecimento intensivo – *knowledge intensive* –, isto é, que exige vários especialistas experientes). Cerca de 70% dos BPR com este objectivo radical falharam (Bashein *et al.*, 1994), no essencial por:

1. Falta de empenhamento e interesse da administração em implementar os novos processos;

2. Âmbito e expectativas do trabalho à partida irrealistas;

3. Resistência à mudança e desadaptação tecnológica da administração e quadros técnicos.

Há, portanto, várias metodologias possíveis para um trabalho de BPR que equilibram o desenho de uma nova organização "ideal" com a situação real da empresa, isto é, que correspondem a uma perspectiva intermédia à puramente incremental e à exclusivamente radical. A tabela nº 5.1, a seguir, apresenta uma metodologia típica [URL-5].

Tabela 5.1 Exemplo de uma metodologia para um BPR intermédio a uma abordagem incremental e a uma radical.

1.	Criação de um enquadramento(ambiente na empresa);
2.	Identificação de clientes e suas necessidades/requisitos;
3.	Descrição dos processos existentes(identificando "o quê" e "porquê");
4.	Medição do desempenho dos processos existentes;
5.	Definição de um novo projecto, redesenhando processos, tendo em atenção: - Os interesses dos clientes; - Os interesses dos quadros da empresa; - A concorrência(outras empresas no mercado); - O *benchmarking*; - As novas tecnologias, sobretudo as informáticas; - Os "porquês" dos processos actuais, identificados no ponto 3; - O desempenho esperado, face ao medido no ponto 4;
6.	Implementação do novo projecto, planeado de acordo com a realidade identificada no ponto 3.

A utilização extensiva dos sistemas informáticos permitirá definir uma nova abordagem para a coordenação do trabalho na empresa.

No entanto, repensar os processos empresariais focando os resultados em vez de as tarefas tem uma mais valia evidente e não é obviamente necessário que o novo projecto faça extensivo uso das tecnologias de informação.

O BPR tem evoluído nos últimos anos no sentido de englobar o realismo do que é efectivamente exequível no ambiente existente, combinando benefícios incrementais com uma visão abrangente que lhe é própria (Davenport, 1994). Consequentemente, a metodologia sugerida no quadro anterior será adequada para a generalidade dos trabalhos de BPR no contexto actual das empresas a semelhança da EXI.

5.2.2 Reengenharia de Sistemas de Informação

A Reengenharia de Sistemas de Informação[1] surge hoje em dia como uma nova necessidade decorrente do Reengenharia de Processo de Negócio (RPN) que é um conceito que teve implementação ao longo da década 90.

Secundando LEMOS[2], com a desmaterialização das empresas, onde o *focus* passou a estar centrado nos seus Sistemas de Informação e no valor intrínseco da Informação, a experiência diz-nos que qualquer processo de mudança substancial dos processos de negócio não pode estar desassociado de um processo de Reengenharia de Sistemas de Informação.

No entanto, nem sempre a Reengenharia de Sistemas de Informação (RSI) é uma consequência da Reengenharia de Processos de Negócio. Esta realidade verifica-se, nomeadamente, ao nível do *Enterprises resource planning* (ERP - Planeamento de recursos empresariais), onde factores como necessidade de integração de diferentes sistemas, *upgrade* do hardware, estruturas legadas de programação complexas e obsoletas, código desorganizado resultado de uma manutenção prolongada sem objectivos claramente definidos e a necessidade de uma documentação actualizada, levam as empresas a optarem por uma solução RSI em busca de uma maior optimização dos recursos informáticos, maior portabilidade e um rácio de manutenção custo/benefício mais atraente, atendendo ao facto de 70% dos orçamentos para os Sistemas de Informação é despendido na manutenção dos mesmos.

A experiência revela-nos que projectos de RSI são caracterizados por possuírem contornos de grande complexidade e um grau de contingência muito elevado devido, a maior parte das vezes, aos requisitos do negócio estarem mal definidos, a arquitectura do sistema mal documentada e o modelo

1 Reengenharia de Sistemas de Informação envolve a Reengenharia de Software, Reengenharia de Base de Dados e Reengenharia de Dados.
2 Carlos Miguel Stoffel Lemos - Consultor da Unidade de Negócio de ERP da PT – Sistemas de Informação.

de dados inexistente. Estes são seguramente os grandes obstáculos ao desenvolvimento e implementação de projectos de RSI [URL-1].

De forma a contornar estes obstáculos, ao nível dos ERP's, aplica-se uma metodologia faseada consubstanciada em dois grandes pilares – Reverse e Forward Engineering, entendendo-se com Reverse Engineering o processo de aprendizagem e re-documentação de alto nível de abstracção do sistema legado; e por Forward um conjunto de actividades de redesenvolvimento e implementação dos novos requisitos [URL-1].

Desta forma a metodologia é genericamente descritiva, baseada em 7 etapas [URL-1]:

Etapa 1: **Definição dos objectivos da Reengenharia e do Negócio**: é uma etapa onde é efectuado o levantamento das necessidades e dos objectivos da reengenharia. Nesta fase é fundamental o envolvimento da direcção do topo (*top management*) para que os objectivos tácticos/estratégicos sejam bem disseminados no projecto de reengenharia.

Etapa 2: **Constituição da equipa de Reengenharia**: a escolha da equipa é de extraordinária importância, visto ser ela que irá estar envolvida ao longo do desenvolvimento, implementação e manutenção do projecto. Deve ser uma equipa integrada e multidisciplinar composta por programadores, utilizadores chaves (*key-users*) e gestores de negócio e do departamento de Sistemas de Informação. É essencial que os utilizadores finais estejam bem representados para que todas as necessidades de informação sejam bem definidas e representativas da realidade operacional. Lemos diz que, segundo a sua experiência, a representação dos utilizadores finais é um factor crítico ou mesmo imperativo de sucesso para os projectos de RSI para evitar-se correr o risco dos projectos se alongarem no tempo com os sucessivos levantamentos das necessidades operacionais e validação dos mesmos.

Etapa 3: **Análise do Sistema Legado**: é uma das fases mais longas do projecto, dado que a maior parte das vezes somos confrontados com uma fraca documentação e mapeamento dos Sistemas de Informação (SI) existentes na empresa. É uma etapa importante, visto ser ela que irá determinar e condicionar a execução da reengenharia.

Etapa 4: **Especificação dos novos requisitos do sistema**: identificados os objectivos e efectuado o re-mapeamento dos programas, dá-se início à especificação funcional e técnica do novo sistema com os elementos funcionais e técnicos do projecto.

Etapa 5: **Definição do processo de implementação**: definidos os objectivos, a equipa e as características do sistema legado, estão reunidas as condições para escolher a estratégia

de implementação. Lemos afirma que, segundo a sua experiência, a inclusão de alterações à especificação ao longo do processo de reengenharia põem em causa todo o trabalho de validação entretanto efectuado. Na opinião de Lemos, qualquer alteração às especificações deve ser efectuado quando o projecto estiver completamente concluído e estabilizado ao longo de pelo menos 8 a 12 meses. O processo de implementação pode ser efectuado segundo duas formas distintas: através de um processo em bloco, onde todas as mudanças ao sistema legado são postas em prática e validadas de uma só vez, ou através de um processo incremental onde à medida que as mudanças são desenvolvidas estas são validadas e colocadas em produção. A escolha do processo de implementação depende da especificidade e risco do projecto em causa.

Etapa 6: **Desenvolvimento e testes**: os testes devem ser efectuados por quem participou no projecto. É uma etapa crítica na medida em que é difícil provar que o novo sistema é funcionalmente equivalente ao software legado. Como foi referido na etapa anterior, alterações às especificações não são aconselháveis, no entanto, caso se verifiquem recomenda-se uma implementação incremental envolvendo os passos ilustrados na figura nº 5.7.

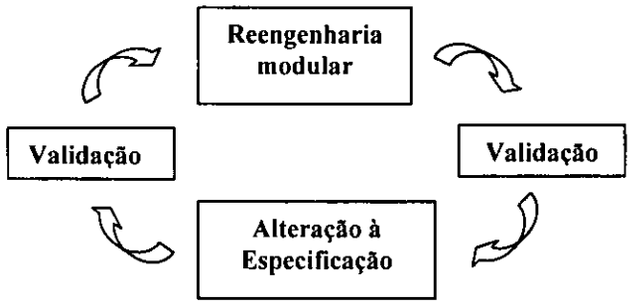


Fig. 5.7: Passos de uma implementação incremental de alterações às especificações [URL-1].

Etapa 7: **Formação:** a formação é uma peça fundamental de consolidação do projecto para todos os utilizadores do novo sistema.

Em suma, o projecto de reengenharia envolve dois grandes componentes: O Reverse e Forward Engineering, sendo o primeiro denominado por Recuperação, o segundo por Redesenvolvimento e a passagem daquele para este constitui a denominada Mudança. Todo este processo de mudança deve estar em conformidade com os objectivos organizacionais traçados conforme ilustra a figura nº 5.8.

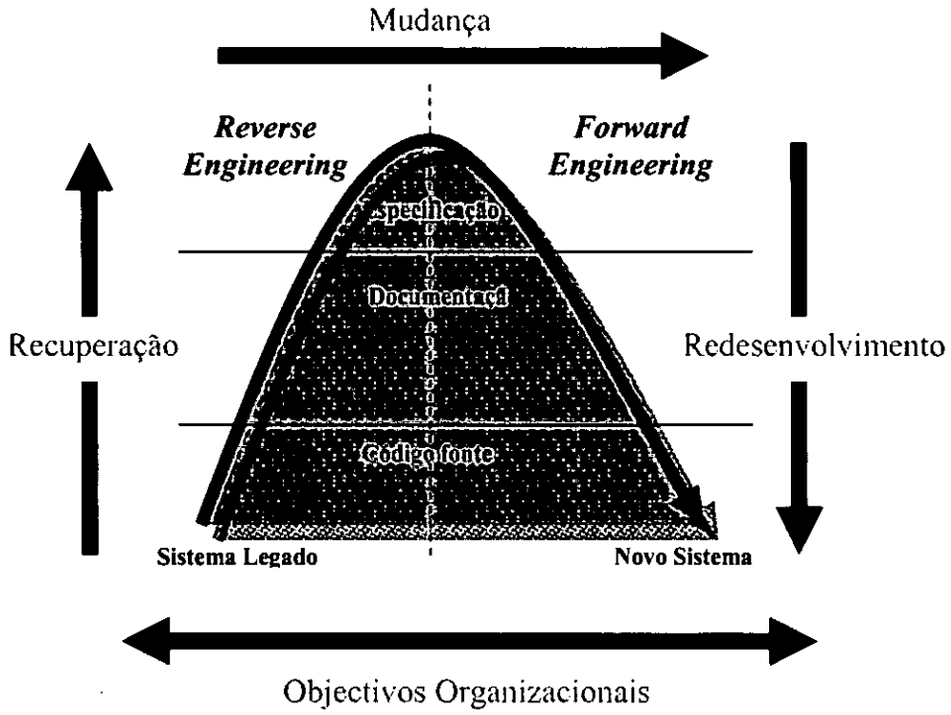


Fig. 5.8: Componentes de um projecto de Reengenharia e o Processo de mudança [URL-1].

A Reengenharia não deve ser uma mera actividade isolada no tempo, pelo contrário, deve ser uma actividade permanente de forma a acompanhar as mudanças das variáveis do negócio e as evoluções tecnológicas. Desta forma, as empresas deverão ter os seus processos mapeados e bem definidos, permitindo que uma dada alteração seja facilmente implementada e sem grande consumo de recursos. A opção de Reengenharia é, na maior parte das vezes, fruto de uma Arquitectura de SI/TI mal definida [URL-1]. Desta forma, antes de se pensar na Reengenharia, as empresas deverão pensar na Arquitectura dos sistemas através da modelização da informação, do mapeamento dos processos de negócio e da documentação dos sistemas e tecnologias existentes na empresa. Dos recursos gastos num projecto de reengenharia, 65% são canalizados para a compreensão e re-documentação dos sistemas existentes. Desta forma, cabe às Direcções de Negócio e Sistemas de Informação das organizações começarem a pensar primeiro na re-arquitectura e em seguida na reengenharia permanente dos Sistemas de Informação, não por uma opção mas pela necessidade de estabelecer uma ordem e controlo dos gastos em recursos afectos aos Sistemas de Informação.

O modelo para uma reengenharia de software não só depende do sistema a modelar como também das ferramentas a usar. O subcapítulo seguinte debruçar-se-á sobre a modelação de um sistema evolutivo, conceitos que podem ser usados no caso específico do Sis-Pública.

5.3 Modelação de Sistemas

A RenaissanceWeb [Renaissance, 1998c], escreve que a modelação de um sistema evolutivo é composta pelos seguintes passos (Fig. 5.9):

- *Modelo contextual do sistema inicial*: Este modelo recebe como entrada toda a documentação existente sobre este sistema e o seu código fonte;
- *Modelo contextual do sistema final ou evolutivo*: A aceitação que o modelo contextual do sistema inicial abrange todos os aspectos necessários para a elaboração do novo sistema, bem como a adição de novos requisitos da organização e de nova tecnologia, resulta no modelo contextual do novo sistema;
- *Modelo Técnico do sistema inicial*: Enquanto que a modelação contextual se preocupa com o que deve ser feito a nível de gestão, o modelo técnico foca em como a funcionalidade deve ser implementada;
- *Modelo técnico do sistema final*: Este é um modelo detalhado, baseado no seu modelo contextual e no modelo técnico do sistema inicial, que permite a adição de novos requisitos técnicos e novas ferramentas ao novo sistema.

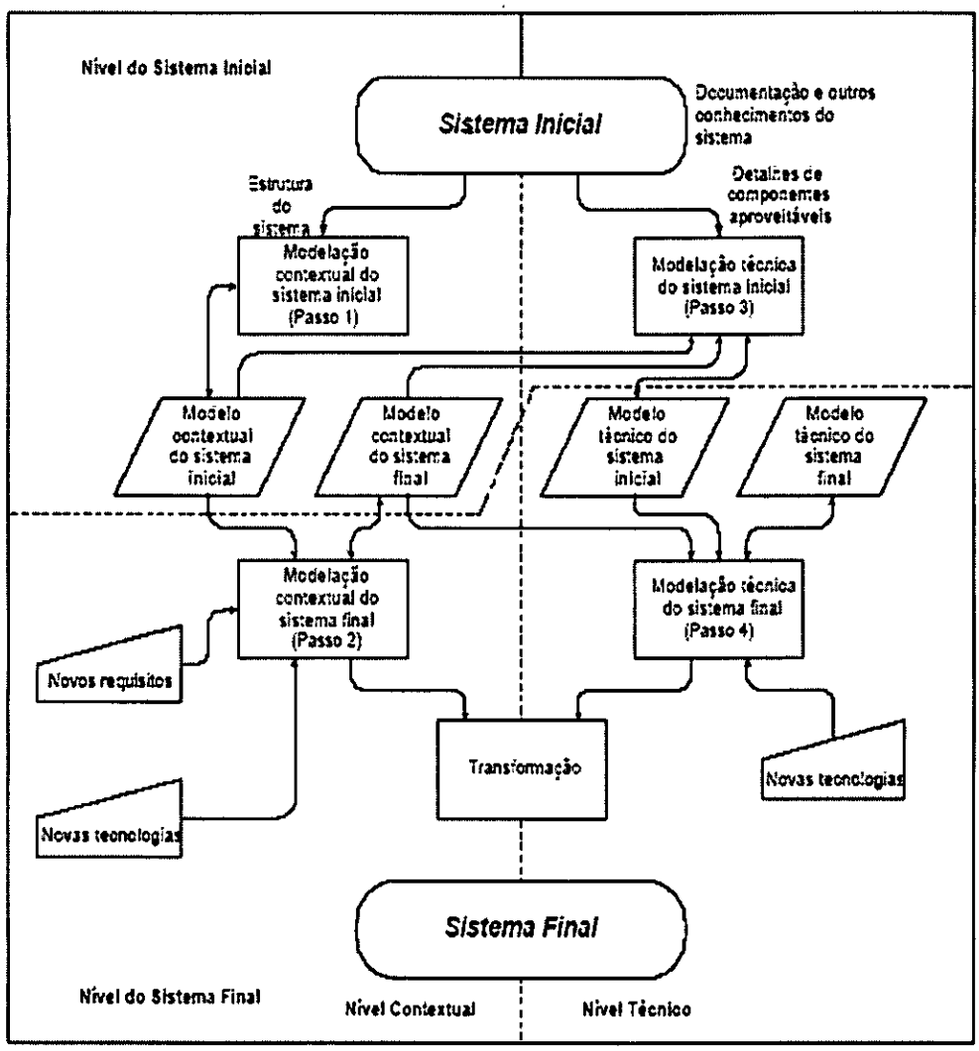


Fig. 5.9 Componentes de uma modelação e sua interligação [Renaissance, 1998c].

5.3.1 Modelação Contextual

Segundo [Renaissance, 1998c], para a evolução de um sistema o modelo contextual deve incluir os seguintes pontos de vista:

- *Ponto de vista do negócio*: Este ponto de vista, está direccionado para os processos de negócio suportados pelo sistema inicial e que deverão ser suportados pelo futuro sistema;
- *Ponto de vista funcional*: É a visão funcional do sistema, ou seja, abrange toda a funcionalidade do sistema;
- *Ponto de vista do ambiente*. Este ponto de vista, é composto pelos pontos de vista lógico e físico do sistema no seu ambiente real de funcionamento.

O ponto de vista lógico ilustra a disposição e interligação lógica entre as diferentes componentes do software do sistema. Por sua vez, o ponto de vista físico ilustra a distribuição destes componentes ao longo do hardware.

5.3.2 Modelação Técnica

Conforme a RenaissanceWeb [Renaissance, 1998c], esta modelação enfatiza o ponto de vista estrutural do software.

Os pontos de vista lógico e físico do sistema no seu ambiente real de funcionamento, referidos na modelação contextual para a evolução de um sistema criam espaço para abordagem, no ponto seguinte, de conceitos de UNIX que por circunstância é o ambiente de funcionamento do Sis-Pública actual e continuará para o novo Sis-Pública.

5.4 Sistemas de informação em ambiente Unix

Referiu-se acima, na definição do problema, que o Sistema Operativo Unix (Ambiente onde funciona o Sis-Pública em Informix 4GL) oferece mecanismos para tornar a vida do administrador ou usuário do Sistema de Informação muito simples e produtiva, e é ainda uma ferramenta de valor inestimável para operações em máquinas remotas. A adicionar, este subcapítulo descreve alguns conceitos de Unix fundamentais na constituição da base teórica e tecnológica para o estudo deste trabalho.

5.4.1 Sessão de trabalho

O uso de UNIX se baseia na noção de *sessão de trabalho*. Cada usuário é designado por um nome-*"login"*, com mais uma senha secreta associada. Uma sessão de trabalho típica consiste das seguintes etapas:

- O usuário identifica-se, fornecendo seu nome-*login* e sua senha ao sistema.
- A sessão de trabalho inicia, com o lançamento do shell (modo texto) ou do ambiente (modo gráfico).
- Uso do sistema (lançamento de comandos e aplicações).
- Fim da sessão (operação de *logout* ou *logoff*).

O UNIX pode gerir diversas sessões simultâneas de usuários distintos na mesma máquina. Cada um terá uma visão independente e transparente dos recursos disponíveis, sem conflitos ou interferências.

5.4.2 Interfaces Gráficas no Sitema Operativo UNIX

No UNIX a interface gráfica é completamente separada do núcleo do sistema operacional (como a figura 5.10 ilustra) permitindo uma grande versatilidade em relação aos ambientes gráficos disponíveis. A interface gráfica é construída em dois níveis[URL-2]:

- O servidor gráfico *X-Windows*, que oferece as funcionalidades gráficas básicas, gere entidades básicas como regiões de tela e trata eventos relacionados à interface (operações de *mouse* e teclado);
- O ambiente de trabalho, composto por vários processos, que implementam a decoração das janelas, menus, ícones, *desktops* virtuais, e etc.

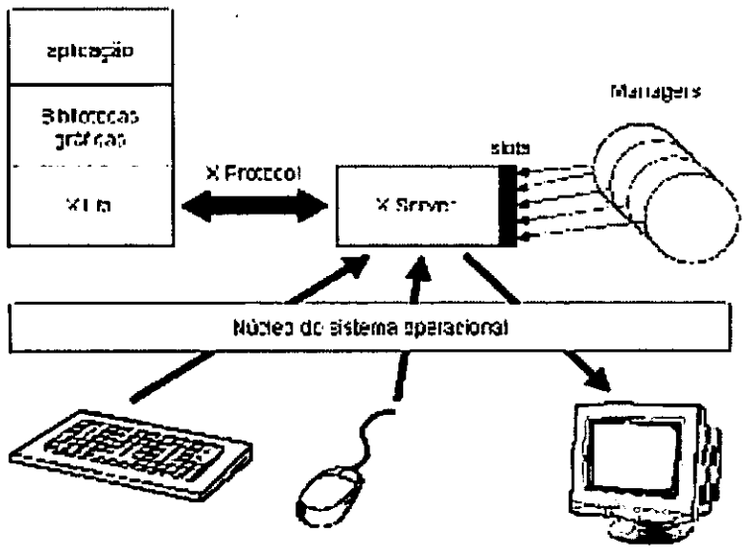


Fig. 5.10: Arquitectura gráfica do UNIX [URL-2].

Algumas características importantes diferenciam a arquitectura gráfica do UNIX daquela implantada em outros sistemas[URL-2]:

- O **ambiente gráfico é implementado fora do kernel**. Com isso eventuais falhas na interface gráfica não comprometem a estabilidade do sistema;
- A comunicação entre o servidor gráfico, o ambiente e as aplicações é feita através de **sockets TCP/IP**, usando um protocolo especial chamado **X-Protocol**. Com isso é possível estabelecer conexões gráficas entre aplicações e servidores gráficos em máquinas separadas;
- Os processos que implementam o ambiente de trabalho são executados com o identificador do usuário (UID), com isso várias sessões gráficas distintas podem ser suportadas na mesma máquina, em *displays* distintos;
- Existem dezenas de possibilidades de ambientes gráficos distintos, à escolha do usuário. As aplicações gráficas irão executar sobre todos eles, sem necessidade de versões específicas para um ou outro ambiente.

Um ambiente gráfico usado é KDE (The K Desktop Environment), e roda sobre um processo servidor X-Windows padrão [URL-1].

À manter a plataforma UNIX e simultaneamente acrescida da componente *Graphical User Interface* (GUI – Interface de Utilizador Gráfico) que pode rodar somente na plataforma WINDOWS está-se perante o uso de plataformas com clientes heterogéneas, aludindo, por constituição, a necessidade de esclarecimento de conceitos de arquitectura Cliente/Servidor à fazer no ponto seguinte.

5.5 Arquitetura Cliente/Servidor

A arquitectura Cliente/Servidor vem sendo desenvolvida há vários anos, porém em pequenos passos. Primeiro, a realocação de aplicações em *Mainframe* para as chamadas plataformas abertas rodando Sistema Operacional UNIX. Depois, em relação a abordagem dos dados, saindo de Sistemas de Arquivos ou Base de Dados Hierárquicos localizados em *Mainframe* para Sistemas de Base de Dados Relacional, e posteriormente, a importância da capacidade gráfica dos pacotes de "front-end" existentes, facilitando a interacção com o usuário (MCKIE,1997).

A arquitectura *Cliente/Servidor* é hoje uma das mais utilizadas em ambientes corporativos, substituindo a arquitectura muito rígida que eram os sistemas envolvendo *mainframe* [URL-6].

Em ambientes corporativos, o compartilhamento de dados era resolvido através da utilização de *mainframe* com vários terminais interligados à eles [URL-6]. Esta estrutura, além de ser muito cara, era muito rígida.

Com o aumento do poder de processamento dos micro-computadores, os fabricantes de programas para micro-computadores começaram a desenvolver banco de redes cada vez mais poderosos, sistemas operacionais mais rápidos e flexíveis, redes locais (LAN - Local Área Network) e redes amplas (WAN - Wide Area Network). Esta arquitectura mostrou-se mais flexível devido a utilização dos micro-computadores em rede, cada vez mais complexas e versáteis, com o compartilhamento de recursos de cada uma das máquina [URL-6].

Este subcapítulo tem por objectivo descrever algumas características técnicas que envolvem a Arquitectura Cliente/Servidor, a sua concepção teórica e a implantação da comunicação entre Cliente e Servidor.

Importa adiantar e destacar o aspecto referente a divisão de tarefas entre o Cliente e o Servidor que melhor depende de cada aplicação em si. Se o Servidor for apenas um Sistema Gestor de Base de Dados, deixando para o Cliente o resto do processamento, ou se algumas tarefas como o de controlo de acessos, análise de dados, e validação dos comandos é executada pelo Servidor, esta é uma decisão do construtor do Sistema em função das características do negócio do usuário. Estas diferenças são mostradas no tópico a seguir.

No contexto do presente trabalho pretende-se utilizar as características do processamento distribuído. Este tipo de processamento apresenta duas configurações para uma arquitectura Cliente/Servidor. A primeira, que é representada por três camadas, é responsável pela visualização da interacção entre os aplicativos e o hardware, como pode ser visto na figura 5.11.

A segunda representação, também fornecida em três camadas, mostra como é tratada a divisão da funcionalidade de uma aplicação, segundo as configurações do *Gartner Group*, como pode ser visto na figura nº 5.12.

Processamento Cliente-Servidor

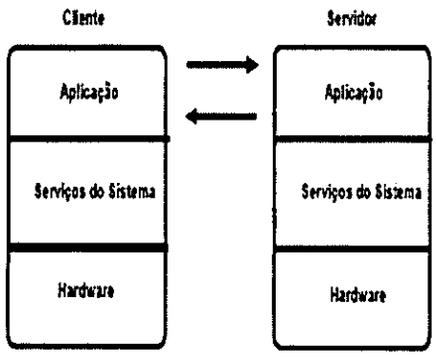


Fig. 5.11 Sistema Cliente/Servidor [URL-6].

Modelo de Distribuição de Processos

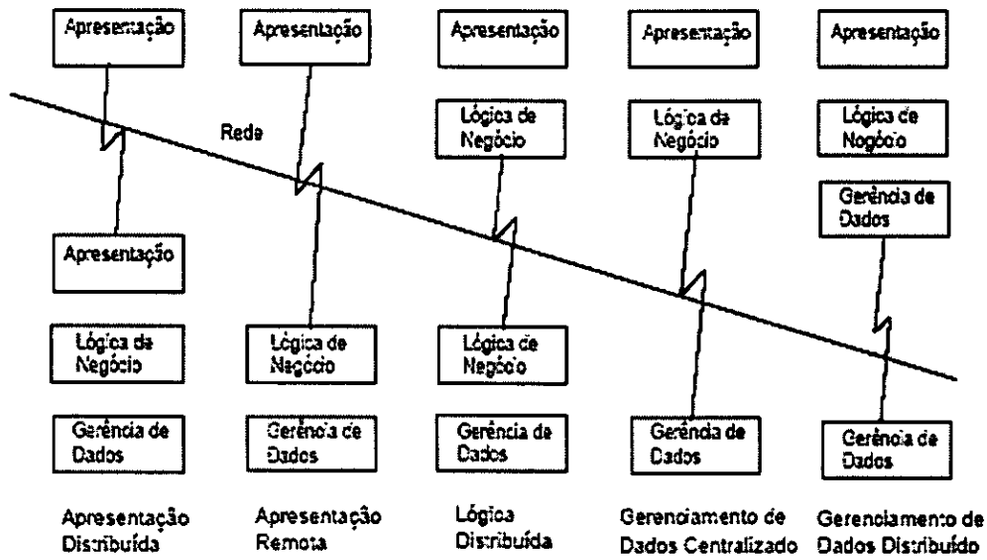


Fig. 5.12. Arquitectura da Aplicação Cliente/Servidor [URL-6].

Dentre as muitas vantagens da arquitectura Cliente/Servidor, pode-se citar, segundo (SALEMI,1993):

• **Confiabilidade**

Se uma máquina apresenta algum problema, ainda que seja um dos Servidores, parte do Sistema continua activo;

• **Matriz de Computadores agregando capacidade de processamento**

A arquitectura Cliente / Servidor provê meios para que as tarefas sejam feitas sem a monopolização dos recursos. Usuários finais podem trabalhar localmente;

• **O Sistema cresce facilmente**

Torna-se fácil modernizar o Sistema quando necessário;

• **O Cliente e o Servidor possuem ambientes operacionais individuais / Sistemas Abertos**

Pode-se misturar várias plataformas para melhor atender às necessidades individuais de diversos sectores e usuários;

Além destas vantagens, segundo (MCKIE,1997), pode-se encontrar dentro de uma arquitectura Cliente/Servidor a interoperabilidade das estações Clientes e Servidoras entre as redes de computadores, a escalabilidade da arquitectura visando o crescimento e a redução dos elementos constituintes, a adaptabilidade de novas tecnologias desenvolvidas, a performance do hardware envolvido na arquitectura, a portabilidade entre as diversas estações que compõem a arquitectura e a segurança dos dados e processos.

Embora o avanço da arquitectura Cliente/Servidor tenha trazido uma variada gama de facilidades para o desenvolvimento de aplicações distribuídas, também possui algumas desvantagens tais como:

• **Manutenção**

As diversas partes envolvidas nem sempre funcionam bem juntas. Quando algum erro ocorre, existe uma extensa lista de itens a serem investigados.

• **Ferramentas**

A escassez de ferramentas de suporte, não raras vezes obriga o desenvolvimento de ferramentas apropriadas, que vai sendo ultrapassada, devido ao grande poderio das novas linguagens de programação, esta dificuldade está se tornando cada vez menor;

• **Treinamento**

A diferença entre a filosofia de desenvolvimento de software para o micro-computador de um fabricante para o outro, não é como a de uma linguagem de programação para outra. Um treinamento mais efectivo torna-se necessário;

• **Gestão**

Aumento da complexidade do ambiente e a escassez de ferramentas de auxílio tornam difícil a Gestão da rede.

5.5.1 Sistema de Três Camadas para a Aplicação

Os Sistemas Cliente/Servidor têm sido utilizados basicamente nos Sistemas de Base de Dados Distribuídos, Processamento de Transações e nos Sistemas de Suporte a decisão. A maioria das aplicações tidas como críticas têm permanecido num *mainframe*. Pode-se definir como críticas aquelas aplicações cujos resultados são utilizados para decisões estratégicas e que, portanto, variam de empresa para empresa, dependendo do seu negócio [URL-6]. Quase que universalmente, os sistemas de finanças são considerados críticos para todas as empresas e os sistemas de processamento de transações são críticos para as companhias de aviação [URL-6].

As aplicações de processamento de transações, apesar de serem tipicamente consideradas como críticas, têm sido frequentemente migradas para a arquitectura Cliente/Servidor, através do processo de "*Downsizing*", como por exemplo os sistemas de entrada de pedidos, pontos de vendas e sistemas de reservas [URL-6]. As transações ocorrem nos Clientes e são formatadas e enviadas ao Servidor para armazenamento. Seria razoável imaginar que as primeiras aplicações a serem desenvolvidas em Sistemas Cliente/Servidor seriam relativamente simples, garantidas e não críticas. Porém, as empresas que se decidiram pelo modelo Cliente/Servidor testaram-no com aplicações de processamento de transações, e na maioria dos casos ficaram satisfeitas com os resultados. Como pode ser visto na figura nº 5.12, o Gartner Group apresenta cinco maneiras de se implementar a arquitectura Cliente/Servidor. Cada camada é dividida entre a parte lógica e a parte de gestão. A primeira representação refere-se a distribuição da camada de Apresentação.

5.5.2 Comunicação entre Cliente e Servidor

Normalmente várias perguntas são feitas para se tentar entender como clientes e servidores se comunicam, que tipo de protocolo é utilizado, e como um cliente encontra o servidor desejado dentro de uma rede de computadores. Embora estas perguntas possam parecer muito difíceis, na realidade elas são relativamente fáceis.

Existem vários conceitos e aspectos comuns às comunicações Cliente/Servidor necessários ao entendimento do funcionamento da comunicação. Nesta secção será dada alguma visão sobre aspectos de conexão.

Para já, importa entender que as redes locais (LAN - Local Area Network) caracterizam-se como sendo redes que permitem a interconexão de equipamentos de comunicação de dados numa pequena região. Costuma-se considerar pequena região distâncias entre 100m a 25km. Existem outras características típicas encontradas e comumente associadas a uma LAN, tais como: Altas taxas de transmissão (até 1Gbps), baixas taxas de erro, geograficamente limitadas e as topologias mais utilizadas são barramento, estrela e anel. Os dois maiores tipos de LAN são: Rede Ponto-a-Ponto (tipo de rede que interliga os Servidores e Clientes no funcionamento do Sis-Pública) e Rede Baseada em Servidor. *Workgroup*, como também são conhecidas as redes ponto-a-ponto, têm algumas vantagens, especialmente para pequenas empresas que não querem fazer um grande investimento em hardware e software do servidor.

5.5.2.1 Conexão TCP/IP

Para este desenvolvimento, a plataforma a ser utilizada é o Windows e Unix para *"front-end" e "back-end"* respectivamente, que apresentam a robustez necessária para os processos multitarefas. A escolha destes ambientes também se deve ao facto deles suportarem vários tipos de protocolos para a comunicação, entre eles o TCP/IP (QUINN,1996).

O Servidor deve ser sempre colocado "no ar", antes de qualquer tentativa de comunicação. Quando a função *Connect* for activada pelo Cliente, uma solicitação de conexão para o Servidor é enviada e, caso a resposta a esta solicitação seja afirmativa, o Cliente e o Servidor passam para o estado de *conectado*.

O capítulo seguinte apresenta um conjunto de ferramentas de migração de interfaces que consubstanciam uma solução se utilizadas com as tecnologias referidas nos capítulos anteriores, e também apresentam-se argumentos da escolha de uma das ferramentas a empregar no caso de estudo.

6. FERRAMENTAS DE MIGRAÇÃO

Com o objectivo de preservar 100% das aplicações em Informix 4GL, incluindo as famosas "customizações", e permitir que se abandone um ambiente de caracteres, sem *mouse*, que é executado num terminal, ou servidor, para uma aplicação com total funcionalidade em ambiente gráfico, entram os serviços de transformação de código, *alavancando* e tornando aberto todo o investimento feito no passado, através de ferramentas de migração (*midleware*).

Obviamente, há um grande investimento de tempo e dinheiro nas aplicações em Informix 4GL. Estas aplicações foram desenvolvidas por valorosos profissionais nas empresas e muitos dos quais já se foram. Por outro lado, essas aplicações embutem lógicas e regras de negócio que não podem ser perdidas, ou que custariam fortunas para serem refeitas por novos profissionais.

6.1 FREEDOM

A ArtinSoft criou a "Freedom", uma ferramenta (tecnologia) capaz de converter até 95% do código automaticamente - algo que é inédito na história do software. O engenho usa uma técnica proprietária ("grammar based reasoning") que rapidamente converte aplicações Informix 4GL em Java, permitindo que se abandone o ambiente de caracteres, sem *mouse*, que é provavelmente executado num terminal, ou num servidor, para uma aplicação com total funcionalidade em ambiente gráfico - que pode ser rodado num ambiente cliente-servidor ou na Web, sem a necessidade de re-treinar os usuários [URL-3], como ilustra a figura nº 6.1.

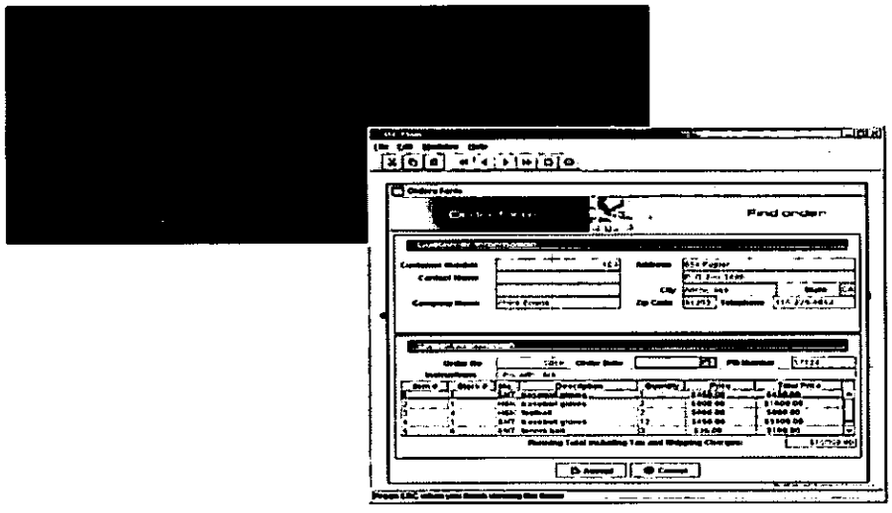


Fig. 6.1: Aplicação no modo texto e no modo gráfico [URL-3].

Isto significa que pode-se redireccionar esforços de desenvolvimento para capitalizar novas oportunidades de mercado e, assim, gerar mais receitas.

Esta ferramenta processa todo o código fonte do Informix 4GL (módulos e forms). A migração automática gera o código Java completamente legível, compilável e executável. Programadores em Informix poderão reconhecer no código Java, a estruturas de comando Informix, nomes das variáveis, comentários e nomes das rotinas, o que facilita a vida dos desenvolvedores originais do I4GL que se tornam familiarizados com o novo código Java. O código pode ser editado em qualquer ferramenta de desenvolvimento JAVA de alto desempenho [URL-3].

A seguir na figura nº 6.2 mostra-se que entradas a ferramenta requer e que saídas o processo tipicamente gera:

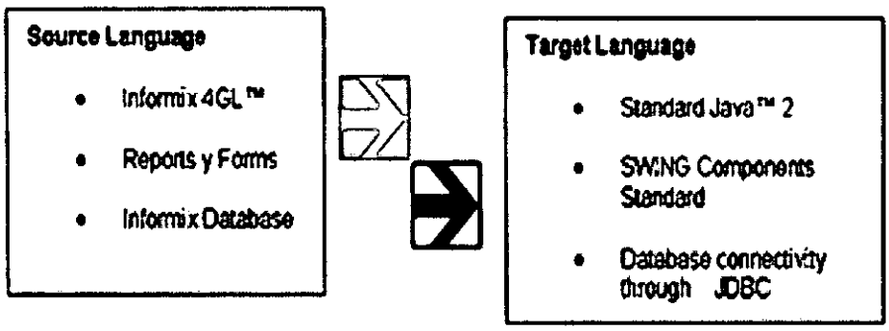


Fig. 6.2: Entradas que a ferramenta requer e saídas que o processo tipicamente gera [URL-3].

Como pode ser visto, o que a ferramenta de migração requer em termos de entrada consiste do código fonte programado em Informix 4GL. Ele deve conter os programas, relatórios, formulários e as estruturas das bases de dados que são usadas na aplicação. Uma vez que a entrada é fornecida, a ferramenta de migração aplica uma série de regras de conversão e gera um código fonte que é 100% compatível com Java 2. Este código fonte usa componentes para a apresentação gráfica da aplicação e a conectividade com bases de dados através de JDBC (*Java Database Connectivity*).

6.2 INFORMIX DYNAMIC 4GL

A partir da primeira versão do Informix 4GL (I4GL), lançado pela Informix Software em 1986, várias foram as organizações que implementaram os seus sistemas usando esta linguagem. Devido à crescente procura de aplicações que suportem a interface gráfica (GUI), em 1998, a Informix Software desenvolveu a ferramenta Informix Dynamic 4GL (D4GL) que permite a migração do interface, baseado em caracteres (UIBC), dos sistemas implementados em I4GL para GUI, sem afectar a sua funcionalidade [Informix, 1999].

Existe a versão 3.01 desta ferramenta que permite uma migração rápida, fácil e que preserva tanto o investimento organizacional do código fonte como o do perfil dos seus técnicos. O resultado de uma

compilação do D4GL pode ser implementado em plataformas clientes tais como Windows (3.11,9x e NT), X Windows (X11) e Web (Java e HTTML) [Informix, 2000].

O D4GL possui um conjunto de variáveis que permitem determinar se a plataforma cliente suporta ou não o GUI, permitindo assim o uso de plataformas clientes heterogéneas. Isto é, num mesmo sistema poderemos ter tanto clientes que suportem GUI como os que somente suportem UIBC. Esta facilidade permite que as organizações possam substituir o seu hardware obsoleto de forma progressiva [Informix, 1999].

6.3 FOUR J'S BUSINESS DEVELOPMENT LANGUAGE (BDL)

Este ponto apresenta as características da *Four J's Business Development Language*(BDL), mencionando algumas diferenças entre o compilador padrão 4GL e Four J's BDL.

6.3.1 A Interface Window

A GUI é exposta num Cliente através dum Servidor Gráfico que pode ser X11 ou versão de Windows pertencente a software Tcl/Tk, chamado WFE (Windows Front End) ou em versões mais recentes WTK (Windows TK) que é providenciado com a Four J's BDL.

Adicionalmente as rotinas de 4GL padrão, as aplicações podem ser expostas com um GUI usando extensão da Four J's BDL – por exemplo para Check Boxes ou List Boxes podem ser acrescidas ao GUI pelo aumento do código de fonte 4GL corrente.

6.3.2 A Interface Web

Pode-se providenciar uma interface Web (*Front End Web*) para aplicações através de um Web Browser compatível, como navegador *Internet Explorer* ou *Netsacape*, usando tanto HTML ou Java para tal. Para usar HTML, precisa-se do demónio Fou J's *HTML Front End*(HFE) que converte o *output* do programa para HTML. E para usar Java, precisa de um *Web Server* que suporta *Servlets* e Four J's *Java Front End*(JFE).

Para usar um dos métodos acima descrito o código fonte 4GL não precisa de ser convertido.

6.3.3 A Interface Baseada em Caracteres

O programa pode também ser executados no modo ASCII (*American Standard Code for Intechange of Information* - Código Padrão Americano para Intercâmbio de Informação). Isto implica que os usuários acedem o seu programa 4GL, através de *logging on* no mesmo computador em que o programa corre. A facilidade da qual pode-se mudar o código permite controlar a taxa de migração do computador Cliente.

6.3.4 Suporte GLS

A Four J´s BDL é suportado pelo Informix Global Language Support (GLS). A configuração GLS permite aos Servidores de Bases de Dados Informix ter acesso a diferentes linguagens, convenções culturais e um conjunto de código.

Quando a Four J´s BDL estiver completamente submetida com Informix GLS, encontram-se algumas restrições. Não pode-se usar GLS com Four J´s BDL e cliente em HTML ou X11.

6.3.5 Arquitectura Cliente/Servidor De Três Camadas

Por uma simples recompilação, pelo Four J´s Universal Compiler (UC), as aplicações podem ser disponibilizas em Sistema Cliente/Servidor de Três Camadas, como mostra a figura nº 6.3, facilitando a migração de terminais baseadas em caracteres para um sistema com interface gráfica de Utilizador (GUI).

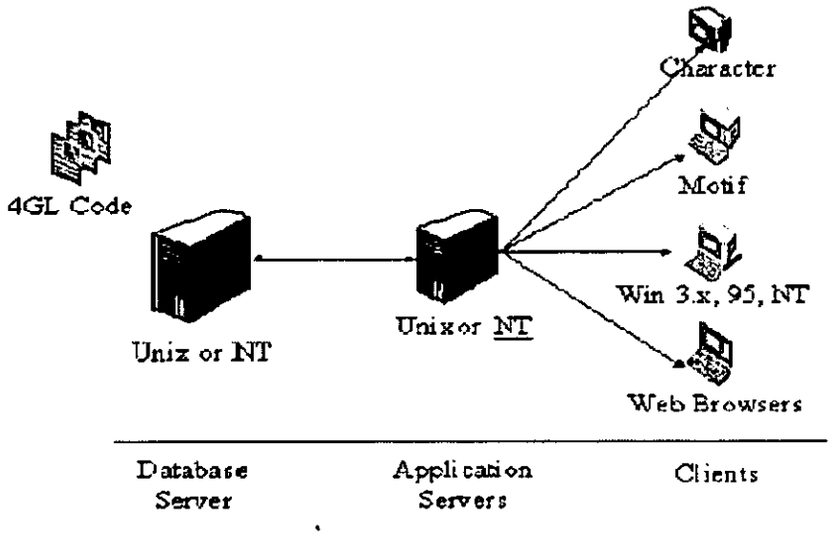


Fig.6.3: Arquitectura Cliente/Servidor de três Camadas [URL-4].

A Four J´s BDL usa esta arquitectura da seguinte maneira:

- A aplicação 4GL é compilada em código P e usada eficientemente num servidor de aplicações onde está instalada a *Dynamic Virtual Machine(DVM)*.
- A Four J´s BDL pode ter acesso a um servidor de base de dados em qualquer ponto da rede.
- Four J´s BDL pode ser acedida pelo ASCII, X11 ou cliente Windows.
- Um servidor de Internet pode ter acesso á Four J´s BDL.
- Os clientes Java ou HTML podem aceder á Four J´s BDL através de *Web Server*.

Exemplo de uma Interfaces de um Computador Cliente

A figura nº 6.4 mostra uma aplicação I4GL antes de ser convertida, numa terminal emulada.

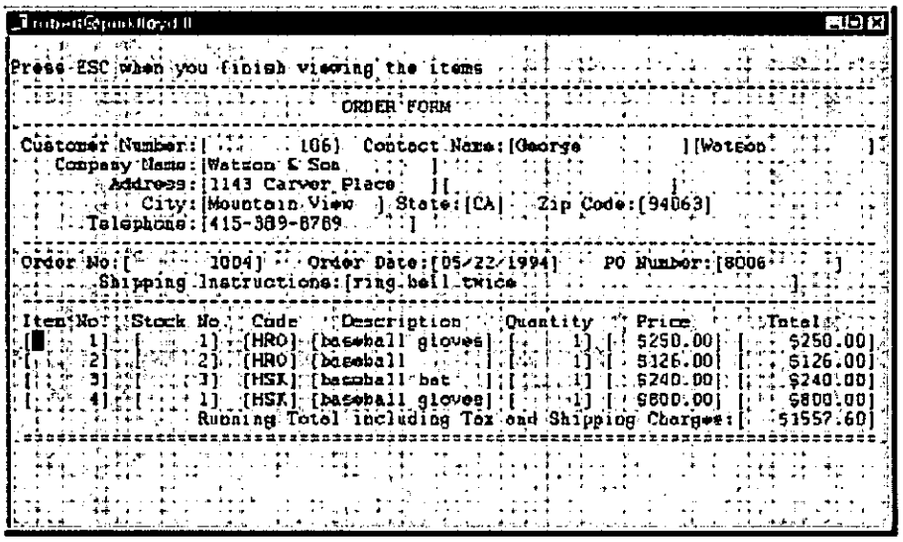


Fig. 6.4: Uma aplicação no modo texto [URL-4].

A Figura nº 6.5 mostra uma aplicação 4GL depois de convertida, num cliente windows.

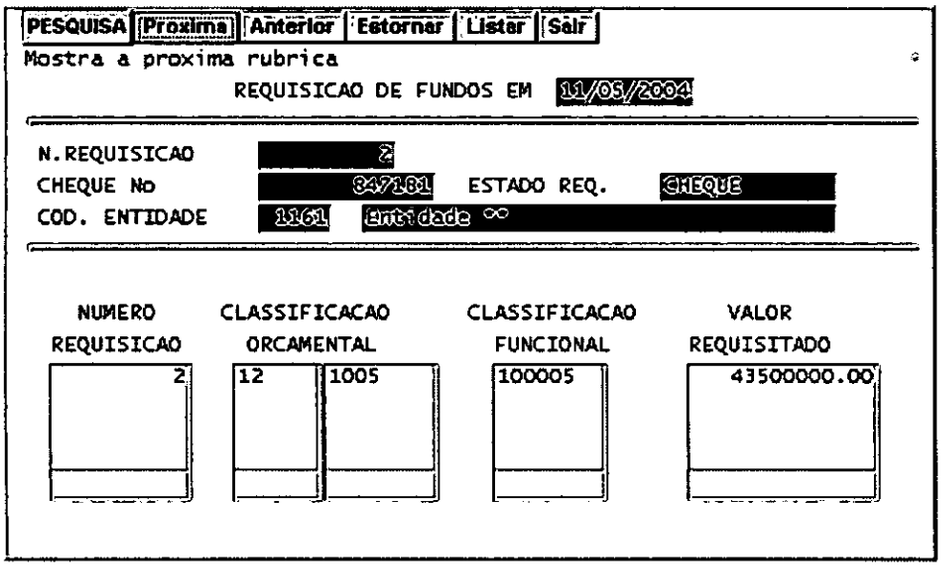


Fig. 6.5: Aplicação no modo gráfico.

6.3.6 A Arquitectura

O Servidor de aplicações pode ser um sistema UNIX ou um sistema Windows NT/2K. Numa instalação típica, a Four J's DBL é instalada num servidor de aplicações (quer na versão de desenvolvimento ou *runtime*) com programas em 4GL. O servidor de base de dados é usualmente instalado no mesmo computador, mas isso não é obrigatório.

O computador cliente pode ser submetido a interface X11 ou computador Windows. Cada cliente tem o seu Four J's BDL *daemon* que carrega a GUI com aspecto de aplicações 4GL.

Ligação Multi-Base de Dados (ODI – *Open Database Interface*)

A interface ODI (*Open Database Interface*) permite as aplicações 4GL ligação aos seguintes motores de base de dados não-Informix: Oracle, IBM DB2, Microsoft SQLServer, Adabas D, PostgresSQL e MySQL [URL-4].

6.3.7 Interface de Base de Dados Dinâmica

Com a ferramenta *fglmkrun* um runner pode ser criado com uma interface de base de dados *estática* ou *dinâmica*. *Runners* criado com interface estático são estaticamente *linkados* com as bibliotecas de interface do sistema de base de dados em causa e podem apenas ligar-se a esse SGBD. *Runners* criados com a interface dinâmica podem carregar dinamicamente em tempo de execução (*runtime*) os drivers de bases de dados, de acordo com a parametrização definida no FGLPROFILE. Com *runners* dinâmicos, programas 4GL podem conectar-se simultaneamente a diferentes motores de bases de dados (por exemplo: Informix e Oracle), com a instrução "CONNECT TO" [Ribeiro e Alexandre, 2004].

6.3.8 *Debugger* Interactivo (*Interactive Debugger*)

O Four J's BDS tem um *debugger* integrado (ao contrário do Informix 4GL em que o *debugger* tinha de ser adquirido em separado). O *debugger* é *parte do sistema de runtime*, pelo que podemos efectuar a depuração das aplicações nos clientes finais, se tivermos o *sources* da aplicação disponível [Ribeiro e Alexandre, 2004].

6.4. ESCOLHA DA FERRAMENTA DE MIGRAÇÃO

Que ferramenta usar?

Após análise das três ferramentas descritas acima e considerando os objectivos deste trabalho, escolheu-se a Four J's BDL.

Esta opção convida os consumidores de sistemas em I4gL, baseado em caracteres, a migrarem para uma nova versão que suporta GUI, como também, permite implementar a opção 4[3] da estratégia de migração.

[3] Veja o ponto 5.1.1.1.Migração da Interface

A Four J's BDL apoia os desenvolvedores de aplicações à adaptar as aplicações existentes em I4GL para a nova economia, suportando integrações sem emendas com os mais recentes padrões tais como HTML, Java, WAP/WML e XML [URL-4].

A utilização da Four J's BDL ajuda a alcançar rapidamente os objectivos de desenvolvimento de aplicações com investimento reduzido no custo e no tempo, e com maior Retorno do Investimento (ROI-*Return on Investment*). Isto porque mantém o código fonte na sua forma original em I4GL, o que reduz custos de aquisição ou recrutamento de novos peritos em outra linguagem de programação, como também não é necessário criar diversas linhas de manutenção, uma vez que o código fonte não altera.

As interfaces das aplicações compiladas em Four J's BDL são heterogéneos, dependendo do hardware do cliente, este poderá suportar ou não GUI. E uma vez compiladas, em Four J's BDL, as aplicações são totalmente portáveis através das plataformas Linux, Unix e Windows, bem como liberta a camada aplicacional da camada de dados transformando a arquitectura do sistema em cliente/servidor de três camadas [URL-1].

7. CASO DE ESTUDO

O presente capítulo aborda o processo de reengenharia do Sistema de Informação para a Gestão de Finanças Públicas (Sis-Pública), sistema crítico para qualquer organização sendo que o conhecimento do negócio, suas regras e procedimentos estão nele embebidos.

Para tal, é utilizada a evolução, a reengenharia do sistema, conceitos e técnicas para o processo de modelação da aplicação, cobertos nos capítulos anteriores.

Pretende-se, com este trabalho, providenciar uma Interface Gráfica aos utilizadores do Sis-Pública. É aplicada a Migração da Interface, estratégia que envolve a actualização ou substituição da interface do utilizador com o sistema, decorrente da Reengenharia de *Software*, uma aproximação ao sistema evolutivo, uma vez que os custos e os riscos de substituição do sistema são elevados.

7.1. Motivação para o estudo do caso

Com as novas tecnologias que surgem para auxiliar ou aprimorar o trabalho do homem, a maioria das aplicações utiliza a famosa interface gráfica (GUI) e algumas já estão evoluindo o conceito, deixando de ser voltadas à aplicação para serem voltadas ao objecto, utilizando a *Object User Interface* (OUI) e paralelamente, com a crescente utilização da Internet, surgiram a *Web User Interface* (WUI) amplamente defendidas dentro da *World Wide Web* (WWW).

A EXI - Engenharia e Comercialização de Sistemas Informáticos possui um sistema de informação para a gestão de finanças públicas, denominado Sis-Pública, desenvolvido na Linguagem Informix 4GL, adoptado por parte da UEM, do INCAJÚ e do PODE, adaptado às novas e modernas correntes da Administração Pública, com mecanismos que respondem à nova filosofia e funcionalidade das Finanças Públicas, garantindo assim, o cumprimento da intenção do Governo que este sistema permita uma gestão aberta, transparente, rigorosa, baseada na legalidade e orientada para o cidadão. O Sis-Pública tem acções que atendem, em grande parte, os objectivos do SISTAFE (novo sistema integrado de orçamento, programação financeira, contabilidade e controlo interno do Estado de Moçambique), de acordo com a responsabilidade fiscal. Este foi desenvolvido dentro do princípio fundamental que rege, modernamente, o desenvolvimento de quase todo o *software* ("o sistema deve-se encarregar da complexidade das tarefas, ao em vez dos usuários, ao em vez dos usuários").

O Sis-Pública atende aos requisitos pré-determinados, oferece a sua funcionalidade de tal maneira que o usuário, para o qual foi planificado, está capaz de controlá-lo e utilizá-lo sem constrangimentos sobre suas capacidades e habilidades. No entanto, este funciona num ambiente com interface baseado em caracteres. Há notável falta de emoção (desagrado) transmitida pelos

utilizadores que trabalham neste ambiente porque a interface não é gráfica, já que estes estão habituados ao ambiente Windows.

Em decorrência disto chegou-se a conclusão que uma boa maneira de minimizar o problema constatado seria a disponibilização da aplicação do Sis-Pública com o recurso interface gráfica nas *front-end*. Este estudo de caso irá basear-se no cenário exposto acima.

7.2 Reengenharia do Sis-Pública

Por forma a contornar os obstáculos e alcançar o objectivo proposto, aplica-se uma metodologia, genericamente descritiva, faseada e consubstanciada em dois grandes pilares – *Reverse e Forward Engineering*. E a fim de iniciar a recompilação deste aplicativo, foram identificados os modelos do sistema legado e sistema evolutivo que são determinados através da modelação do sistema (modelação contextual e modelação técnica) realizada.

Os objectivos do modelo elaborado estão focados para a migração do interface do sistema, mas a utilização da ferramenta seleccionada não só vai satisfazer este requisito como também permite libertar a camada aplicacional, no sistema actual, da camada de dados, modificando implicitamente a arquitectura do sistema, sem alterar o seu código fonte.

Os subpontos seguintes compreendem as 7 etapas da metodologia:

7.2.1 Etapa 1: **Definição dos Objectivos da Reengenharia e do Negócio**

A lista de necessidades e a definição dos objectivos do trabalho constituem o produto desta fase.

É evidente, com o exposto acima deste subponto, a necessidade da EXI acompanhar estas constantes alterações de cenário tecnológico com a rapidez que lhes é exigida, para não ser superada pelas concorrentes. Advém, do referido no período anterior, o objectivo de migrar a aplicação do Sistema de Informação para a Gestão de Finanças Públicas, Sis-Pública, de um ambiente Informix 4GL baseado em caracteres para um ambiente gráfico, utilizando a Four J's *Business Development Language* (BDL), mantendo a funcionalidade existente.

7.2.2 Etapa 2: **Constituição da Equipa de Reengenharia**

A equipa, para o caso do presente projecto, foi constituída pelo autor do trabalho, utilizadores, gestores de finanças públicas e especialistas da ferramenta, com o apoio do pessoal do Departamento de Software Ware (DSW) da EXI.

7.2.3 Etapa 3: **Análise do Sistema Legado**

Esta fase reveste-se de importância no projecto, e não foi longa como o esperado porque contou com o apoio de todos membros da equipa. Os constrangimentos ergonométricos no Sis-Pública são ilustrados no modelo contextual do sistema inicial (sistema antigo).

Modelação do Sistema

SIS-PÚBLICA E MODELAÇÃO CONTEXTUAL

Sistema inicial

Sob ponto de vista dos processos de negócio, o sistema actual, responde correctamente às necessidades dos utilizadores e das organizações. Este sistema é composto por seguintes módulos principais:

- Novo Ano (*novano*): É o módulo responsável pela definição de todos os parâmetros para o funcionamento do Sis-Pública;
- Contabilidade (*contabi*): Este é o módulo centro das actividades do Sis-Pública, responsável pela cabimentação de requisições;
- Gestão (*Gestao*): É o módulo responsável pela definição de operadores, impressoras, terminais, formatos numéricos e menus;
- Tesouraria (*tesou*): Este é o módulo responsável pelas operações de tesouraria executadas pelo Sis-Pública;
- Requisição de Fundos (*reqfund*): É o módulo responsável pela emissão de requisições de fundo;
- Orçamentos (*orcam*): É o módulo responsável pela consulta de dotações orçamentais e funcionais;
- Banco (*banco*): Este módulo é responsável pela definição dos bancos, contas, realização de depósitos, listagens e conferências.

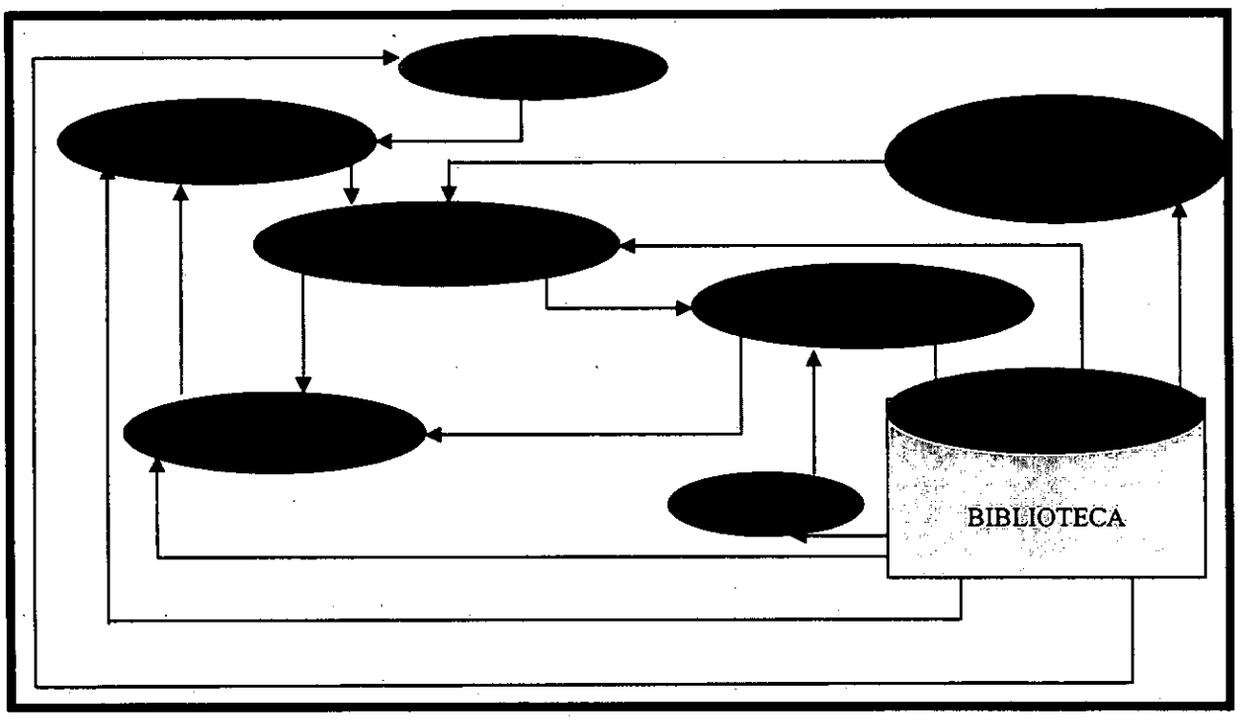


Fig. 7.1: Relação entre os módulos do Sis-Pública.

O Sis-Pública é usado pelos seus utilizadores para as actividades seguintes:

- Os gestores da empresa acedem ao sistema apenas para consultas e para a extracção de determinados mapas;
- Os administradores do sistema ou técnicos acedem ao sistema para parametrização, introdução ou alterações de dados, consultas de dados e extracção de mapas;
- Os utilizadores simples apenas acedem ao sistema para introdução, alteração, consultas de dados e extracção de mapas.

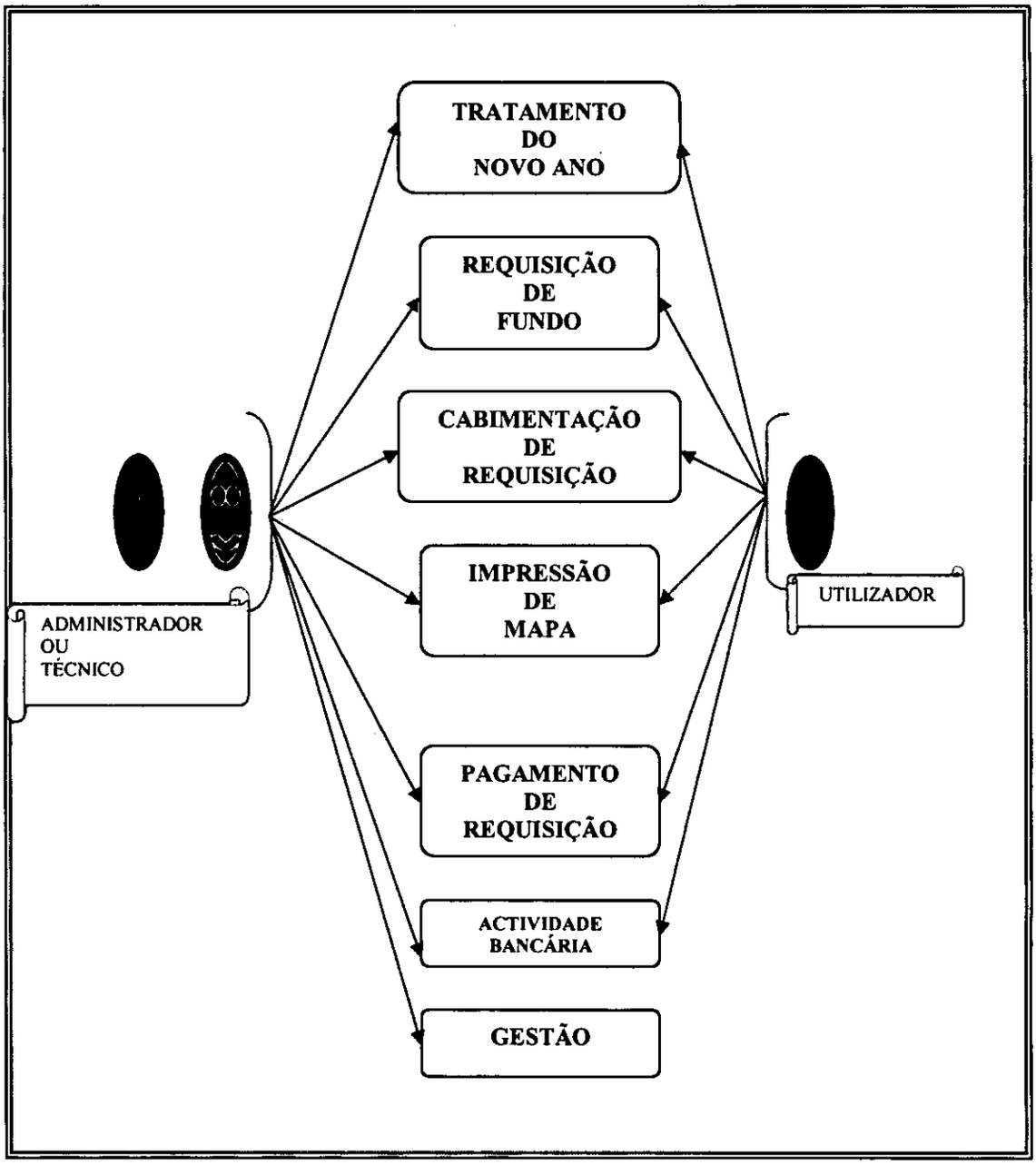
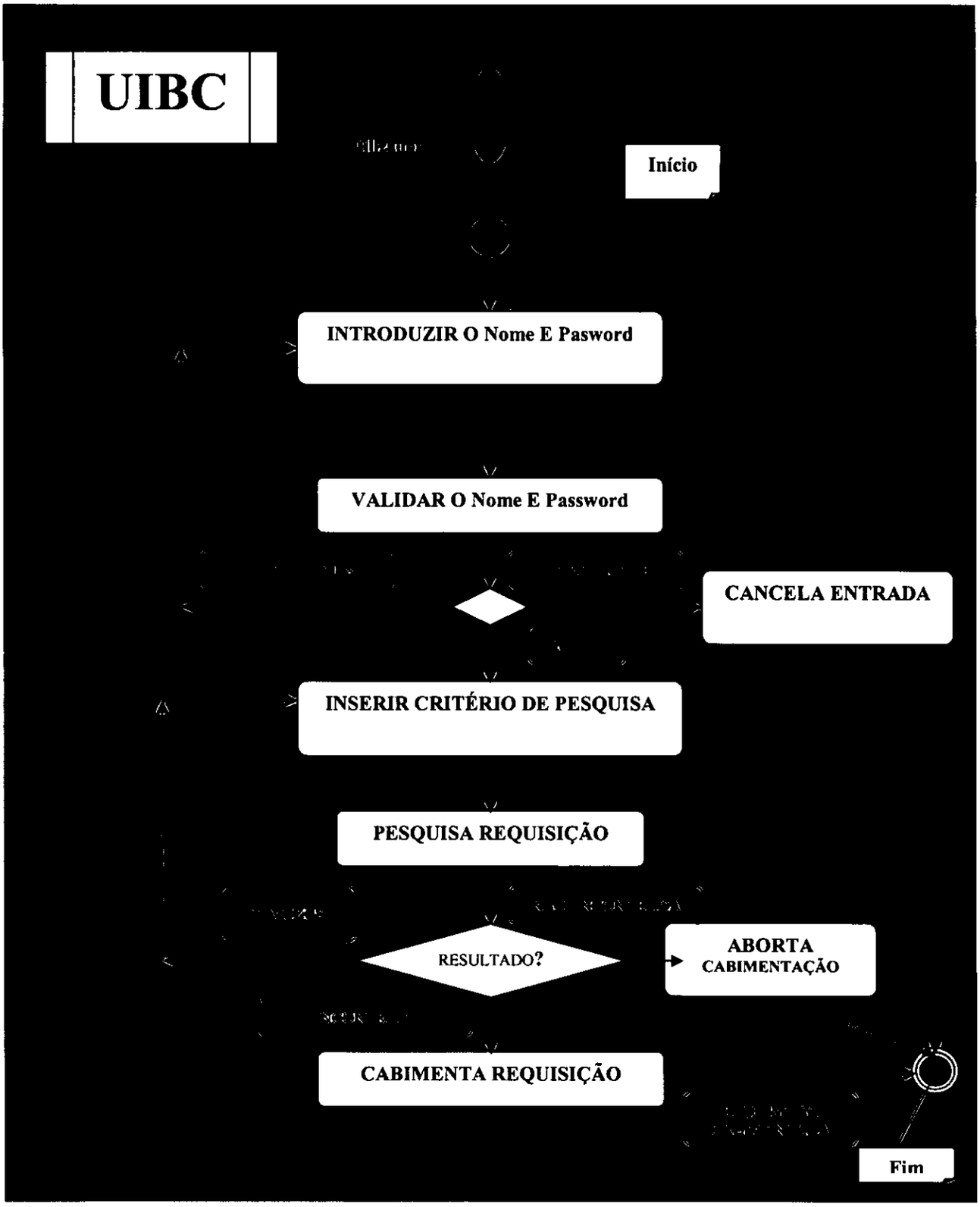


Fig. 7.2: Actividades suportadas pelo Sis-Pública e alguns utilizadores.

A funcionalidade do Sis-Pública não será alterada, tendo em conta o objectivo do estudo que refere a manutenção da funcionalidade existente. A cabimentação de uma requisição ilustra este ponto de vista na figura nº 7.3.


UIBC
Início
INTRODUZIR O Nome E Pasword
VALIDAR O Nome E Password
CANCELA ENTRADA
INSERIR CRITÉRIO DE PESQUISA
PESQUISA REQUISIÇÃO
RESULTADO?
ABORTA
CABIMENTAÇÃO
CABIMENTA REQUISIÇÃO
Fim

Descrição do diagrama de actividades da cabimentação de uma requisição no Sis-Pública

I. INÍCIO;

II. O utilizador introduz o *Nome* e *Password* ;

III. O sistema valida o *Nome* e *Password* introduzido;

IV. Se o *Nome* e *Password* for rejeitado então (II.), em cada uma das duas primeiras rejeições consecutivas; Se o *Nome* e *Password* for rejeitado três vezes consecutivas então (V.); Se o *Login* e *Password* for aceite então (VI.);

V. Cancela-se a entrada ao sistema e segue (X.);

VI. Insere-se o critério de pesquisa da requisição através da interface baseada em caracteres (UIBC), segue (VII.);

VII. Pesquisa-se a requisição segundo o critério introduzido, segue (VIII.);

VIII. Se o resultado for não aceitável então (VI.) ou (IX.); Se o resultado for aceite então (X.);

IX. Aborta-se a pesquisa da requisição por cabimentar e segue (X.);

X. Cabimenta-se a requisição pesquisada, o sistema altera o estado da requisição (T→C) e regista a cabimentação, segue (XI.) ou (VI);

XI. FIM;

Para representar o sistema sob o ponto de vista do ambiente, focam-se os pontos de vista lógico e físico separadamente através de diagramas de blocos.

- O ponto de vista lógico é ilustrado na Fig. 7.4 onde:
  - O sistema está implementado sob arquitectura cliente/servidor de duas camadas;
  - A camada cliente é composta por uma interface remota do sistema, baseada em caracteres (constrangimento ergonométrico);
  - Na camada do servidor estão definidas as regras do negócio, bibliotecas necessárias para o funcionamento do sistema, o sistema de gestão de base de dados e as suas interligações.

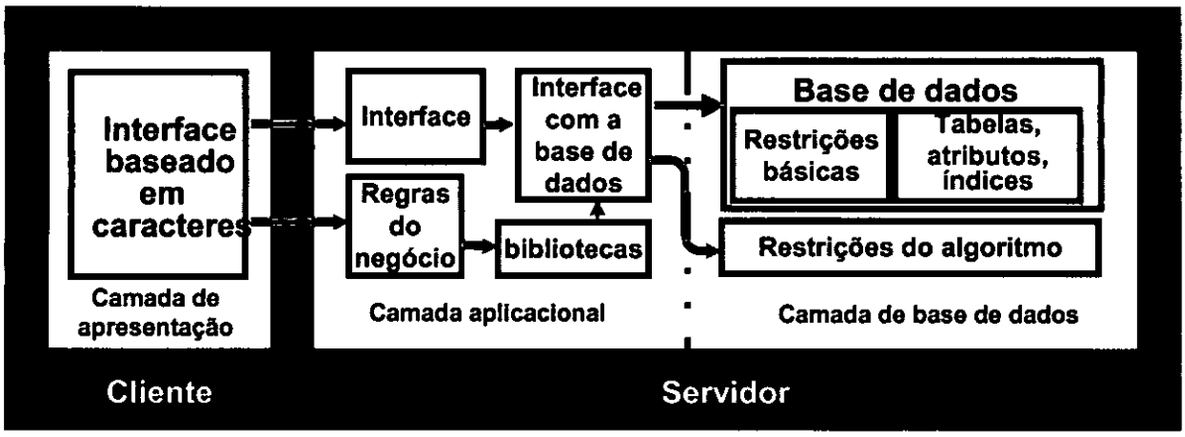


Fig. 7.4: Estrutura Lógica do sistema.

A fig. 7.5 ilustra a distribuição física do sistema actual (Ponto de vista físico), onde se destacam os clientes por se comportarem como terminais não inteligentes e que se comunicam com o servidor através de uma rede local (LAN) usando o protocolo TCP/IP.

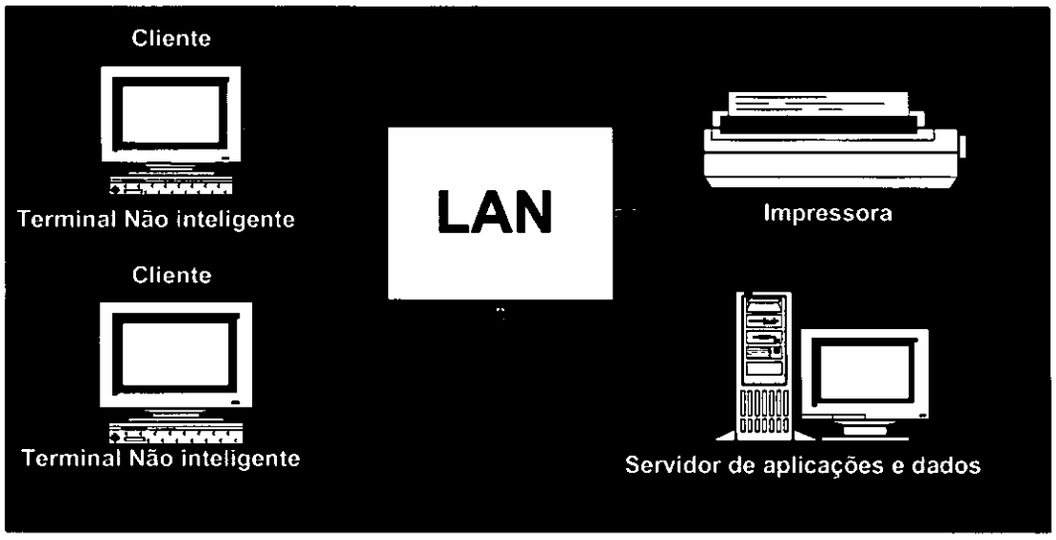


Fig. 7.5: Estrutura física do sistema actual.

7.2.4. (Etapa 4): **Especificação de Novos Requisitos**

Depois de estabelecer os objectivos básicos da migração da aplicação, é necessária a especificação funcional e técnica do novo sistema com os elementos funcionais e técnicos do projecto. O objectivo aqui não é o aprofundamento nos detalhes da implementação de cada recurso da aplicação, mas ao invés disto, demonstrar as actividades e eventos que a aplicação deverá suportar:

- A aplicação deverá aceder a base de dados que estão em arquitectura cliente/servidor *Unix* ou *Windows;*
- Estabelecer um *"Front-End Server"* num computador *Windows* que vai permitir a aplicação assumir tanto o perfil de Cliente como o de Servidor;

- Disponibilizar "*WorkStation's*" com uma Interface Gráfica (GUI), "agradável", para pesquisa, modificação de registos e visualização de relatórios (mapas).

Tendo em vista os tópicos estabelecidos, é demonstrado na figura nº 7.6, o esquema do Sis-Pública que será desenvolvido.

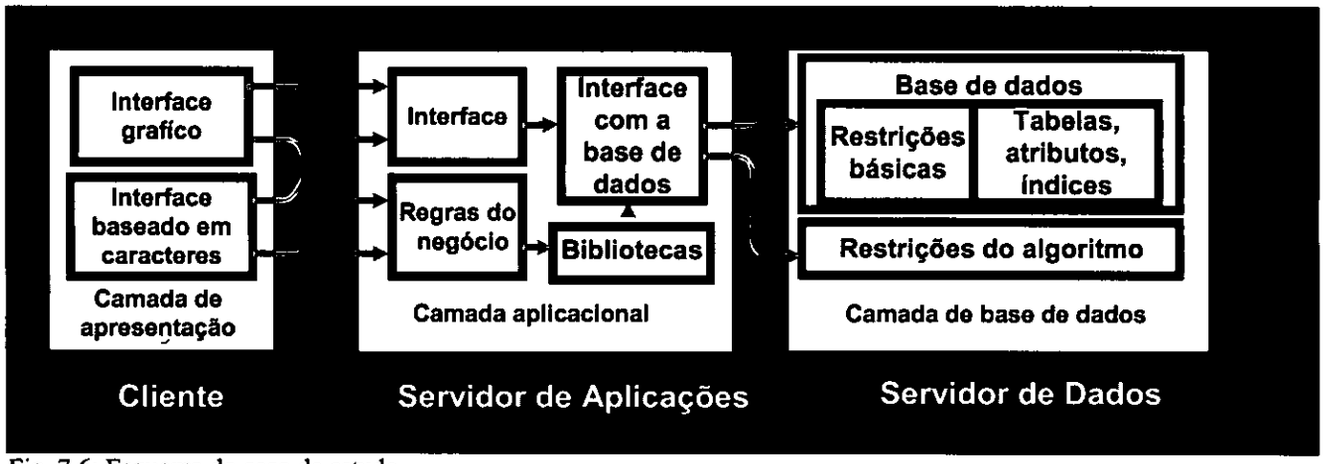


Fig. 7.6: Esquema do caso de estudo.

Ao aceitar que o modelo contextual do sistema inicial abranja todos os aspectos necessários para a elaboração do novo sistema, bem como a adição de novos requisitos da organização e de nova tecnologia, teremos como resultado o modelo contextual do novo sistema.

SIS-PÚBLICA E MODELAÇÃO CONTEXTUAL

Novo sistema

- O ponto de vista de negócio mantém-se idêntico ao do modelo contextual do sistema inicial, uma vez que apenas se pretende alterar o interface do sistema.
- A funcionalidade também não se altera. Usar-se-á o caso anterior (fig. 7.3) para demonstrar, simulando-o no novo sistema, figura nº 7.7.

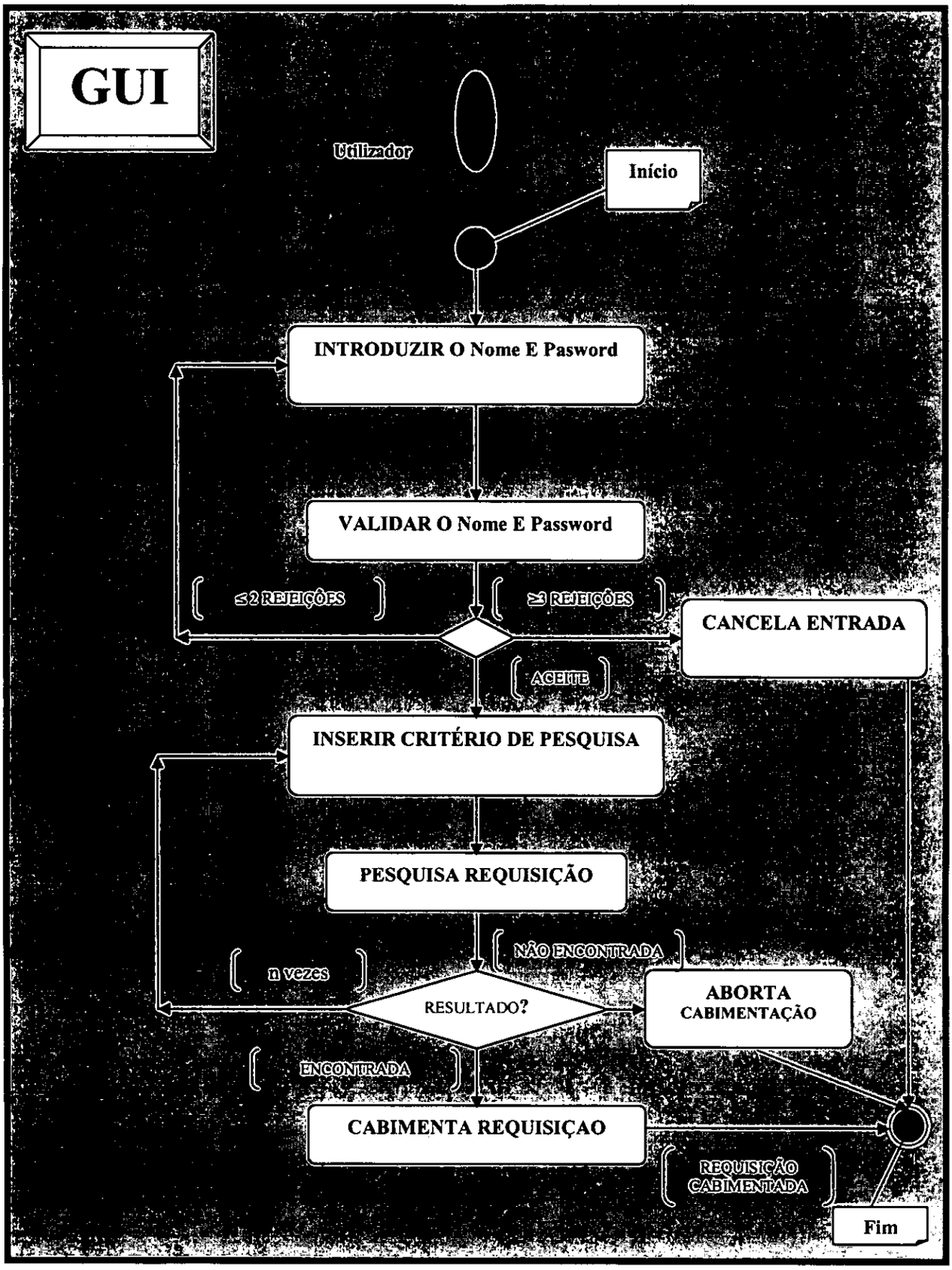


Fig. 7.7: Actividades de cabimentação de uma requisição no novo Sis-Pública.

Pode-se verificar que a funcionalidade do novo sistema é idêntica à do inicial, e que a única diferença é o interface (UIBC e GUI) entre o utilizador e o sistema.

Uma vez que a ferramenta de migração suporta clientes heterogéneos, o modelo contextual do novo sistema sofreu alterações sob o ponto de vista do ambiente, tais como:

- Sob ponto de vista lógico figura 7.8:
  - A camada aplicacional separar-se-á da camada de dados, com possibilidade de tornar a arquitectura do sistema em cliente servidor 3T;
  - O novo sistema suportará clientes heterogéneos, UIBC e GUI.

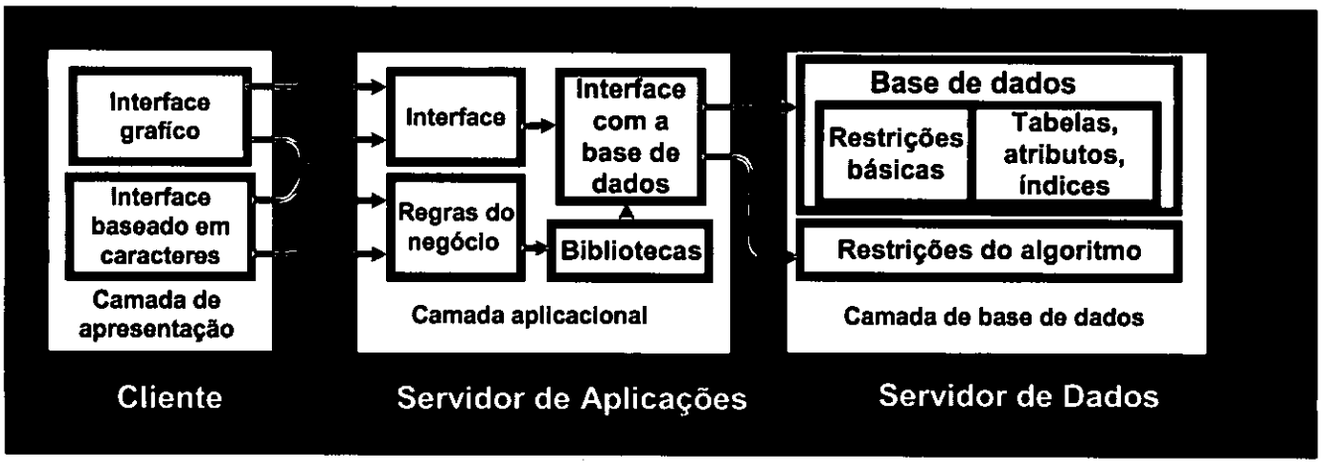


Fig. 7.8: Distribuição lógica do sistema final.

- Sob ponto de vista físico figura nº 7.9:
  - O sistema poderá ter um servidor de aplicação diferente do de dados;
  - Permitirá o acesso ao sistema de clientes que suportem ou não a GUI.

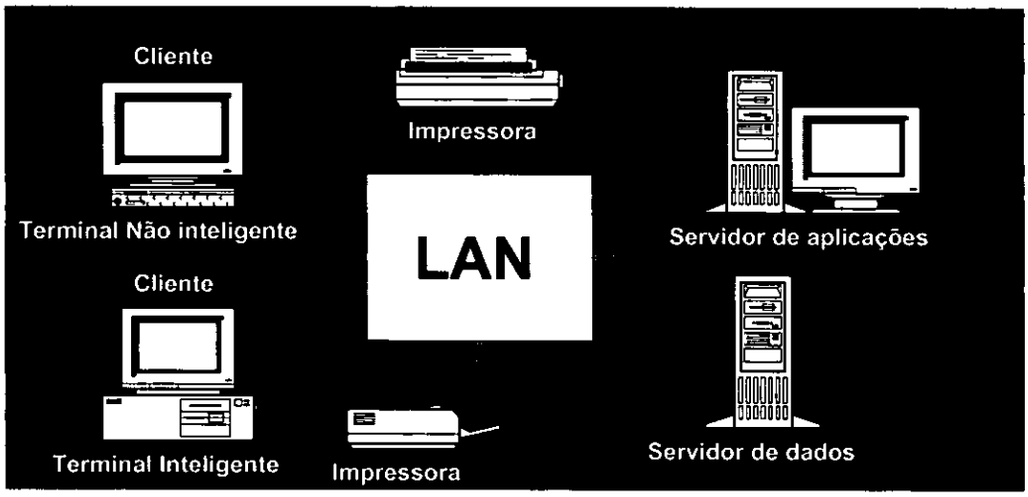


Fig. 7.9: Estrutura física do sistema final.

7.2.5 Etapa 5: Definição do Processo de Implementação

A escolha do processo de implementação dependeu da especificidade e risco do projecto, em causa.

Definidos os objectivos, a equipa e as características do sistema legado, estão reunidas as condições para escolher a estratégia de implementação que pode ser efectuada através de um processo em bloco, onde todas as mudanças ao sistema legado são postas em prática e validadas de uma só vez. Aspectos relacionados a implementação da funcionalidade são vistos, bem como a adição de novos requisitos técnicos e novas ferramentas ao novo sistema baseado no modelo contextual e modelo técnico inicial.

SIS-PÚBLICA E MODELAÇÃO TÉCNICA

A estrutura da geração, em Informix 4GL, de executáveis é ilustrada através das fig. 7.10 e 7.11 para o compilado e para o interpretado respectivamente [Informix, 1994].

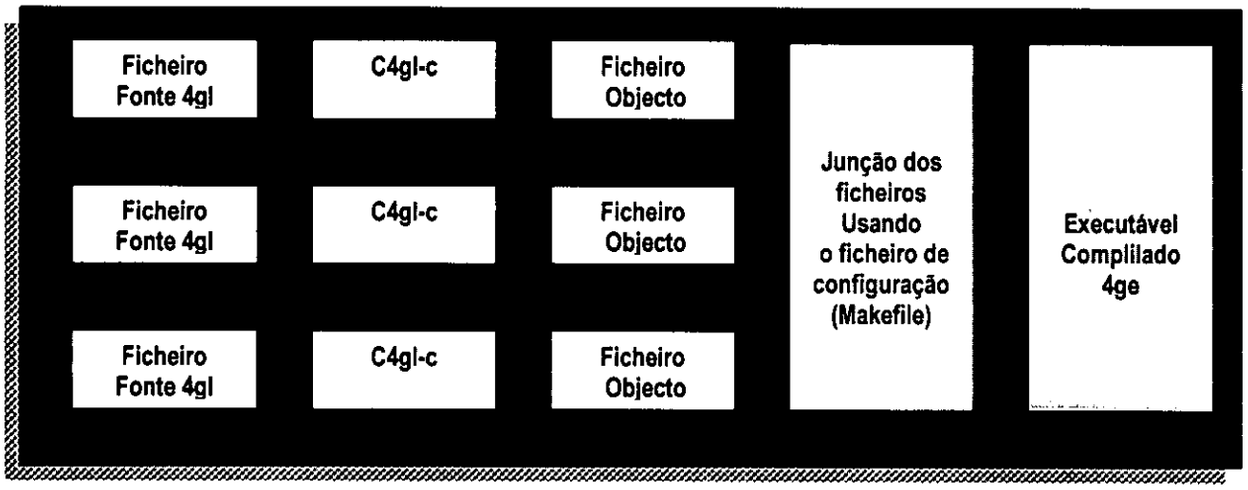


Fig. 7.10: Estrutura da geração, em I4GL, do executável compilado.

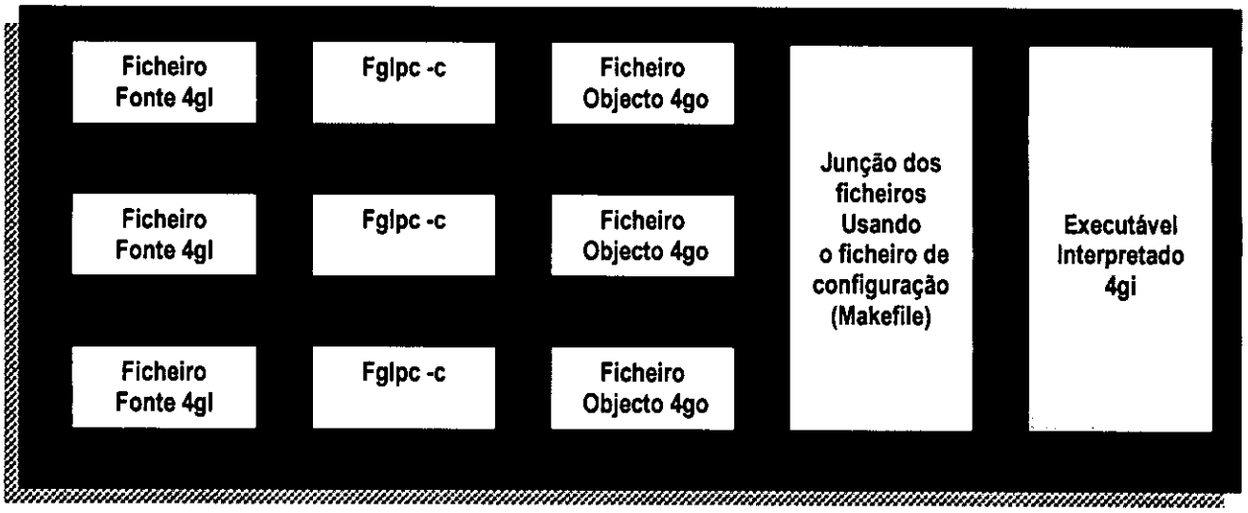


Fig. 7.11: Estrutura da geração, em I4GL, do executável interpretado.

Nesta secção descreve-se mais o modelo técnico do novo sistema focando, principalmente os aspectos que o diferenciam do modelo técnico do sistema inicial, uma vez que a ferramenta de

migração usa os ficheiros fonte com estrutura e sintaxe semelhante aos do I4GL. Desta forma, as propriedades que se destacam encontram-se nas tabelas 7.1e 7.2 [Ribeiro e Alexandre, 2004]:

Tabela 7.1: Convenções utilizadas nas extensões de ficheiros na compilação em C-Code

Extensão	**Descrição**
.4gl	Ficheiros source de Four J's BDL.
.42c	Módulos em C-Code modules resultantes do pré-processamento de BDL para C.
.42o	Objectos compilados resultantes da compilação de C.
.42e	Programa executável resultante da linkagem de C.

Tabela 7.2: Convenções utilizadas nas extensões de ficheiros na compilação em P-Code

Extensão	**Descrição**
.4gl	Ficheiros source de Four J's BDL.
.42m	Módulos P-Code resultantes da compilação.
.42r	Programas P-Code resultantes da linkagem.
.42e	Bibliotecas P-Code resultantes da linkagem.

- A extensão dos formulários compilados é comum a P-Code e C-Code e será **42f**.

Para se ilustrar os procedimentos completos de compilação do *P-code* e o esquema completo de compilação do *C-code*, para a geração dos interpretados e os executáveis, são usadas as figuras 7.12 e 7.13 respectivamente e anexo II. Alguns exemplos de ficheiros em Informix 4GL são mostrados no anexo IV.

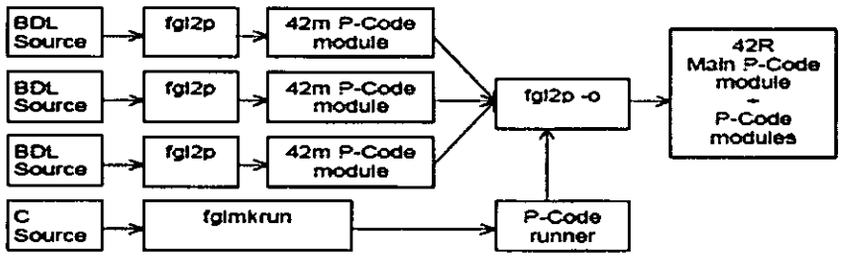


Fig. 7.12: Procedimento de compilação e linkagem em *P-code* - BDL [Ribeiro e Alexandre, 2004].

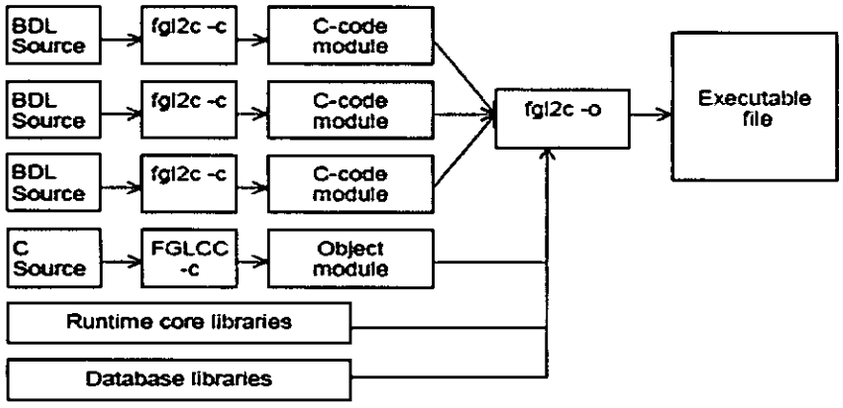


Fig. 7.13: Esquema de compilação e linkagem em *C-code* – BDL [Ribeiro e Alexandre, 2004].

7.2.6 Etapa 6: Desenvolvimento e Teste

Para atender à implementação do modelo de implementação da migração de interface do utilizador e à avaliação da mesma foram efectuados testes pelas entidades que participaram no projecto, com o objectivo de provar que o novo sistema é funcionalmente equivalente ao software legado. As alterações às especificações, que se verificaram, tiveram uma implementação incremental.

7.2.6.1 *Teste do Modelo*

Atendendo a necessidade de alcançar o objectivo desta etapa e, em parte, a necessidade de lançar no mercado o Sis-Pública novo, deu-se lugar ao uso do modelo (e da ferramenta) que se fez valer através de teste do mesmo. Seleccionou-se um dos módulos do Sis-Pública para testar o modelo proposto. O módulo seleccionado foi o "Contabilidade" por ser o módulo centro das actividades do Sis-Pública e que devido à sua funcionalidade pode dizer-se que é o coração do Sis-Pública.

A figura nº 7.14 ilustra a sequência resumida do teste do modelo onde o módulo inicial (com interface UIBC) serve de fonte (*Input*) para o modelo, tendo como resultado o módulo final (com interface gráfica).

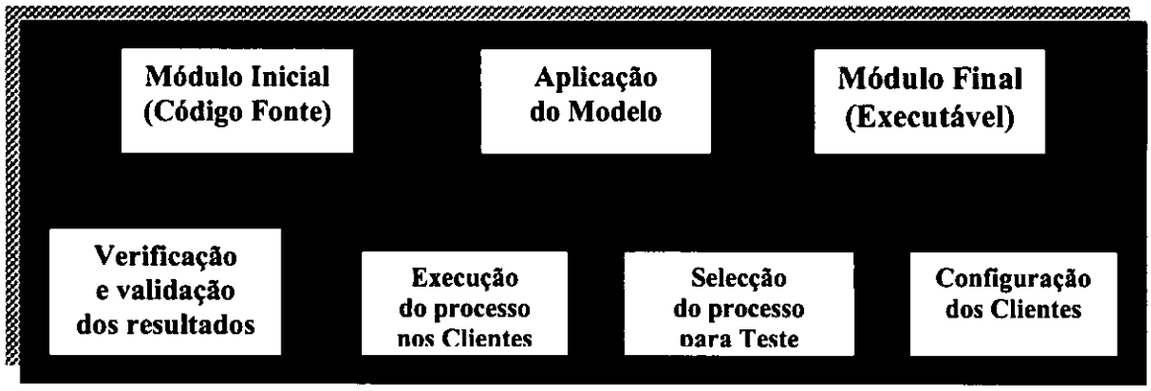


Fig. 7.14: Sequência de compilação e teste do modelo.

Para testar o modelo, as seguintes etapas são definidas:

- Aplicar o modelo ao módulo escolhido;
- Gerar o executável usando o Four J's BDL;
- Configurar os dois tipos de clientes (UIBC e GUI);
- Seleccionar o processo a ser testado;
- Executar o processo seleccionado usando os dois tipos de clientes;
- Verificar e validar os resultados.

A seguir serão ilustrados em detalhe os passos acima referidos:

a) Aplicar o modelo ao módulo seleccionado.

Para testar o módulo seleccionado, primeiro teve de se aplicar o modelo proposto de acordo com as fases da fig. 7.14.

b) Gerar o executável usando BDL.

A compilação e a geração dos executáveis em BDL foi feita de acordo com os comandos e sequências descritas na fig. 7.12.

Tabela 7.3: Actividades para a compilação de P-Code para executáveis em BDL

Actividades	Procedimentos	Resultados
Compilação dos ficheiros dos módulos *sources* em BDL (*.4gl)	For I in *.4gl Do fgl2p –c Si Done Ou Fgl2p *.4gl	Ficheiros compilados dos módulos em *P-code* (*.42m)
Geração de executáveis por *lincagem* (*.42r)	fgl2p –o *.42r *.42m Ou **Nome_módulo.link.4js**	Ficheiro de executável interpretado (*.42r)
Compilação dos formularios (*.per)	For I in *.per Do fglform Si Done Ou fglform *.per ou **mf4 *.per**	Formularios compilados em BDL (*.42f)

c) Configurar os dois tipos de clientes (UIBC e GUI)

Como o cliente UIBC não necessita de configuração, pois usa um emulador de terminal, esta parte do relatório descreve somente a configuração do cliente GUI, onde foi necessário instalar os componentes "I4GL Server" e "Informix Client-SDK". Estes componentes são responsáveis pela conversão dos comandos ou instruções enviadas entre o cliente e o servidor.

Uma vez instalados estes componentes, efectuou-se a configuração do cliente através das variáveis de ambiente:

- **FLGUI=1** " O valor **1** Indica que o cliente suporta a GUI" ou (**FLGUI=0** " O valor **0** Indica que o cliente suporta o UIBC");
- **FGLSERVER="Nome_servidor/ IP"**.

d) Seleccionar o processo a testar

Para dar seguimento ao desenho do modelo deste trabalho usou-se o exemplo de cabimentação de uma requisição para testes.

e) Executar o processo seleccionado usando os dois tipos de clientes.

Para a execução do processo escolhido, foram seleccionados registos de três requisições cativadas, cujos detalhes estão ilustrados no anexo III. A primeira requisição foi introduzida no sistema antigo, a segunda e a terceira no sistema novo, usando os clientes UIBC e GUI, respectivamente (fig. 7.15).

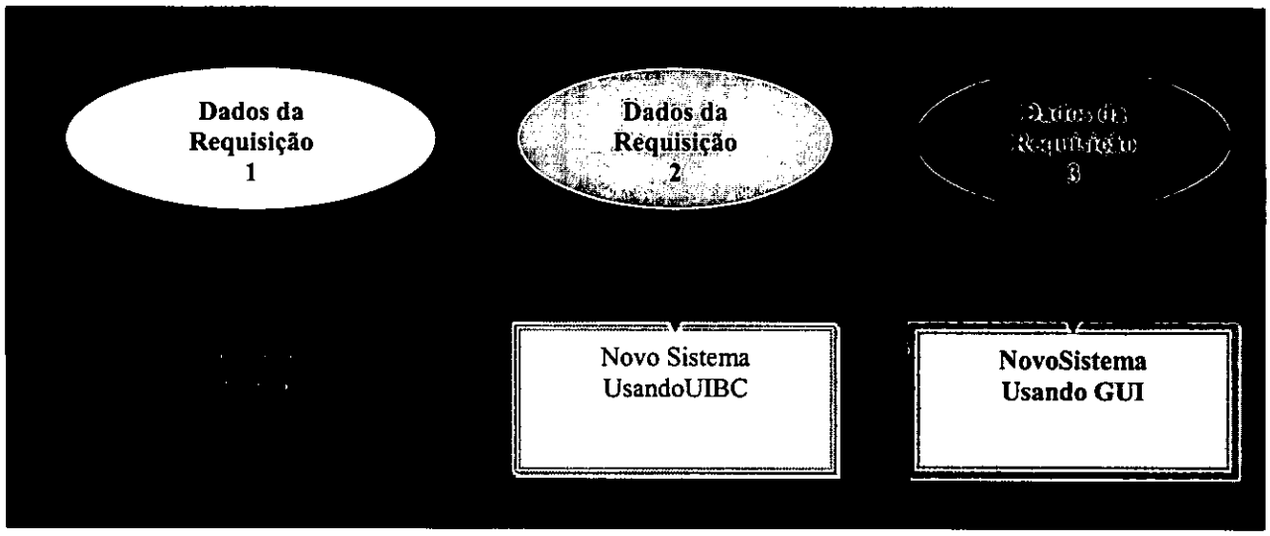


Fig. 7.15: Esquema resumido da cabimentação das três requisições durante o teste.

Os formulários do cliente UIBC são semelhantes aos do antigo sistema, o que facilitou a inserção dos dados para quem está familiarizado com o antigo sistema (seguiram-se os passos da fig. 7.3). A figura nº 7.16 ilustra o formulário de requisição, durante a cabimentação da requisição número 2 a partir de um cliente UIBC.

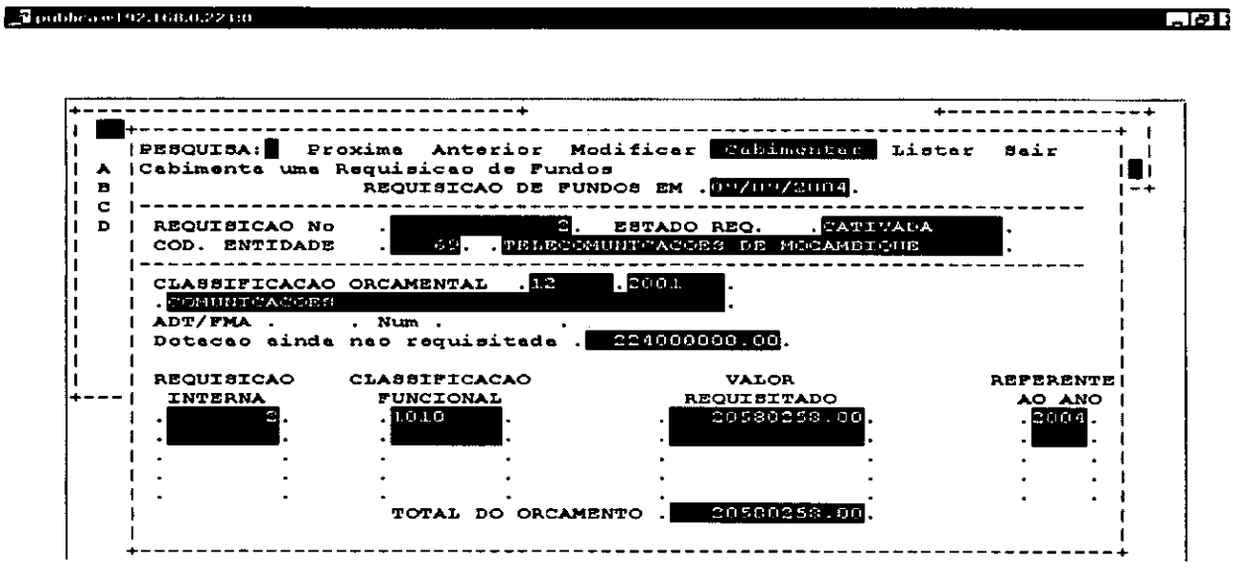


Fig. 7.16: Cabimentação de requisição nº 2 usando o cliente UIBC.

Para o teste do cliente GUI, observaram-se os passos ilustrados na fig. 7.7, onde o novo formulário oferecia, para além do novo visual, possibilidades de se usarem novos dispositivos tais como o rato

para seleccionar os campos a digitar durante a cabimentação da requisição. O uso deste tipo de formulários foi acolhido com agrado pelos utilizadores do sistema antigo. A fig. 7.17 ilustra o formulário de uma nova requisição durante a cabimentação da requisição número 3 a partir de um cliente GUI.

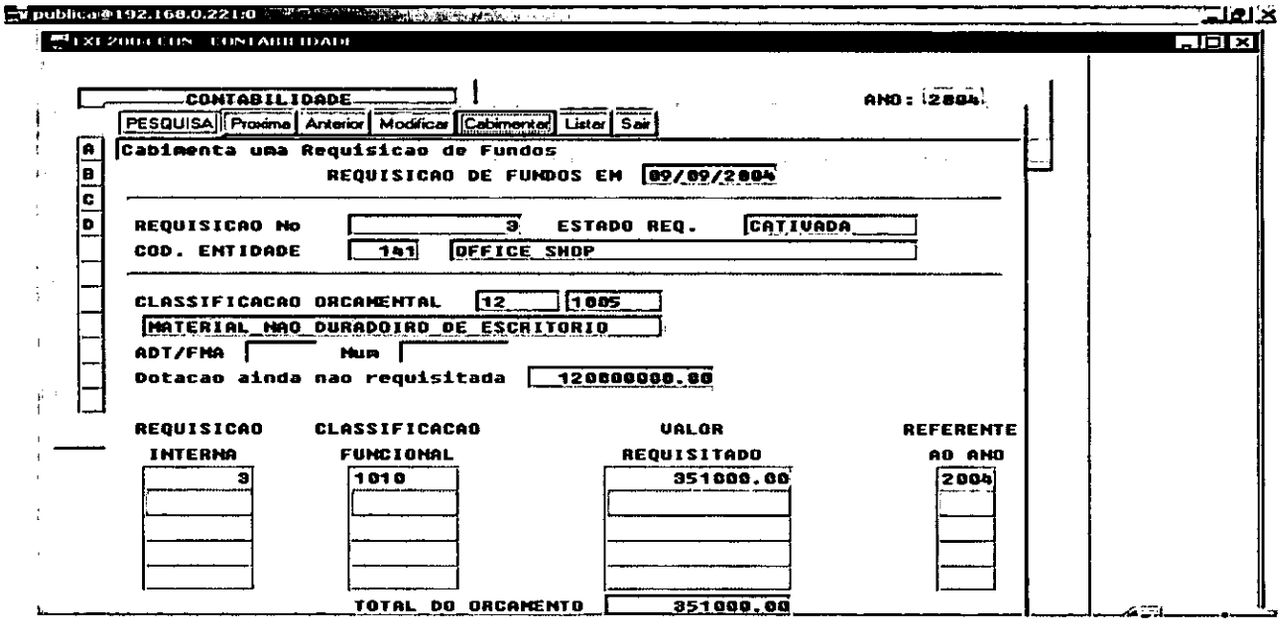


Fig. 7.17: Cabimentação de requisição nº 3 usando o cliente GUI.

f) Verificar e validar os resultados.

Para validar os dados introduzidos, acedeu-se às bases de dados (sistema antigo e sistema novo). Primeiro listaram-se os dados introduzidos na base de dados. Depois, verificou-se a qualidade dos dados resultantes da cabimentação.

Como ilustram os resultados presentes no anexo III, os três registos de requisições cabimentadas comportaram-se como se do sistema antigo se tratasse, validando assim o facto de que a funcionalidade do sistema não foi alterada, apesar da migração da sua interface, o que pode dar uma avaliação (positiva) à implementação do modelo de migração da interface obtido.

7.2.4. Etapa 7: **Formação**

Porque a formação é fundamental para a consolidação do projecto de Sistema de Informação para utilizadores, construiu-se uma lista, relação de necessidades, figuras e descrição de procedimentos à inserir no novo manual do utilizador, tendo em conta que o projecto estará completamente concluído e estabilizado ao longo de pelo menos 8 a 12 meses (altura na qual a alteração às especificações pode ser efectuada na totalidade), embora a funcionalidade do Sis-Pública não tenha mudado e atendendo que a mudança fundamentou-se na interface do utilizador(UI) que era somente

UIBC, passando a GUI. O referido no período anterior, em parte, justifica a continuidade provisória de utilização do manual do utilizador do antigo sistema, que os utilizadores estão sujeitos, bem como o menor esforço empreendido na formação dos utilizadores para trabalharem no Sis-Pública novo que integra o ambiente gráfico (interface de utilizador gráfica).

8. CONCLUSÃO E RECOMENDAÇÕES

A contínua evolução da Informática, tem levado a um progressivo desenvolvimento das capacidades funcionais e de processamento dos computadores permitindo a sua difusão por um maior número de utilizadores. Paralelamente a este desenvolvimento, a Interface com o Utilizador tem evoluído para aproveitar as maiores capacidades de processamento dos computadores no sentido de integrarem mais capacidades humanas.

O modelo desenhado e usado ao abrigo da Reengenharia de Sistemas de Informação, aplicada na parte prática deste trabalho possibilitou:

- a ilustração do constrangimento ergonométrico descrito, sob ponto de vista lógico no modelo contextual do sistema antigo, pela característica da interface do utilizador no Sistema de Informação para a Gestão de Finanças Públicas (Sis-Pública) da EXI - Egenharia e Comercialização de Sistemas Informáticos, cuja camada cliente das aplicações é composta por uma interface remota do sistema em Informix 4GL baseado em caracteres;
- a utilização da Four J's *Business Development Language* (BDL), como ferramenta de migração.

Ao empregar a Four J's BDL e o compilador (Four J's *Universal Compiler*) para providenciar no Sis-Pública uma interface gráfica (GUI) de baixo custo e de relativa facilidade de implementação, solucionou-se o problema de Interface Homem-Computador das aplicações em Informix 4GL (I4GL) baseado em caracteres, que foi o caso de estudo deste trabalho.

A Migração da Interface foi a estratégia da Reengenharia de *Software*, aplicada para que o Sis-Pública da EXI suporte GUI.

A avaliação da implementação do modelo de migração obtido foi via teste e com os resultados obtidos foi possível constatar que os três registos de requisições cabimentadas, a partir dos ambientes baseados em caracteres (sistema antigo e sistema novo) e ambiente gráfico, comportaram-se como se do sistema antigo se tratasse, através da verificação da qualidade dos dados, o que valida deste modo o facto de:

- a funcionalidade do sistema não ter sofrido alteração;
- a ferramenta de migração de interface de utilizador (Four J's BDL) ter atendido ao propósito da sua aquisição (*aplicabilidade*).

Com esta migração de interface do utilizador, foi possível:

- a disponibilização do Sis-Pública com uma interface gráfica(GUI) e uma arquitectura Cliente/Servidor de três camadas, suportando integrações, sem emendas, com os mais recentes padrões tais como HTML, Java, WAP/WML e XML;
- a interoperação do Sis-Pública com várias bases de dados, incluindo Informix, DB2, Oracle e MS SQL, permitindo o entendimento pelos desenvolvedores, da funcionalidade e conectividade das aplicações existentes;
- a criação de desenvolvimentos aplicacionais de forma a que o Sis-Pública opere em ambientes mistos, através do comando *fglmkrun* que pode criar um *runner*, específico de Four J's BDL em P-Code, com um interface de base de dados *estático* ou *dinâmico*;
- o fortalecimento da decisão e da ideia, de migrar as outras aplicações em Informix 4GL que a EXI possui e implementação da componente *Web*;
- a obtenção de garantia de que a EXI e seus clientes não serão obrigados a migrar para um novo sistema, convidando os que desejarem usufruir das novas facilidades da GUI a fazerem a migração para a nova versão.

A particularidade de coexistência dos dois ambientes, baseado em caracteres e gráfico, trabalhando em paralelo com mesma funcionalidade, não forçará a EXI em criar equipas distintas de manutenção, porque o ambiente de manutenção para ambos é o mesmo.

As avaliações constantes, incorporando itens de qualidade, levando em conta os usuários, na medida das expansões e necessidades de mercado, garantem o aumento do ciclo de vida dos sistemas, assegurando sua usabilidade, agradabilidade e melhor razão custo benefício dos investimentos realizados – este é o caso da aplicação (Sis-Pública).

A integração de interfaces baseada em caracteres (UIBC) e gráfica (GUI), permite que as aplicações I4GL alcancem uma maior expressividade a partir das interfaces do utilizador complementares que introduzem *modalidades* redundantes de comunicação Homem-Computador.

No entanto a simples adição de novas *modalidades* não garante, por si só, um aumento da qualidade de uma interface, é necessário que haja uma correcta integração para que a nova *modalidade* traga algum valor acrescentado à qualidade da interface.

A partir deste trabalho é possível recomendar:

- a Reengenharia de Sistemas, aqui usada, para promover uma transformação sistemática dos sistemas de informação de forma a melhorar a sua qualidade, performance, funcionalidade e a

capacidade de evoluir a baixo custo, num curto período de tempo e com menos riscos para os consumidores ou utilizadores finais;

- a Migração da Interface noutras aplicações I4GL, uma vez que os custos e os riscos de substituição do sistema são elevados;
- a implementação de um projecto de integração da componente *Web* no Sis-Pública, atendendo a portabilidade do sistema obtido.

9. REFERÊNCIAS BIBLIOGRÁFICAS

1. (MCKIE,1997) McKie, Stewart "Everything you wanted to know about Client/Server computing but were afraid to ask" Eye on Technology, http://www.duke.com/controller/Issues/DecJan/Clientse.htm

2. (PEDRIANA,1996) Pedriana, Paul "OWLSock - Implementation" <paulp@ccnet.com>, http://users.ccnet.com/~paulp/owlsock/html/download.htm.

3. (QUINN,1996) Quinn, Bob & Shute, Dave "Windows Sockets Network Programming" Addison Wesley, 1996

4. [Bashein *et al.*, 1994] BASHEIN, B.J.; MARKUS, M. L.; RILEY, P., "Preconditions for BPR Success and How to Prevent Failures", *Information Systems Management*, 11 (2), 1994, pp. 7-13.

5. [Da Luz, Chaves e Nunes. 2001] Da Luz ,Jocimar Gomes; Chaves, Neumir Juliano e Nunes, Wasman Ferreira Da Fonseca. EXTRAÇÃO E DISPONIBILIZAÇÃO DE INFORMAÇÕES ENTRE MÚLTIPLAS PLATAFORMAS. Fundação de Estudos Sociais do Paraná - FESP. CURITIBA. 2001. http://celepar7cta.pr.gov.br/Celepar/SiteCel.nsf/0/4bc256bfd9a5232d03256b0d006481b0?OpenDocument.

6. [Davenport e Short, 1990] DAVENPORT, T. H.; SHORT, T. E., "The New Industrial Engineering: Information Technology and Business Process Redesign", *Sloan Management Review*, Summer issue, 1990, pp. 11-27.

7. [Davenport, 1994] DAVENPORT, T. H., "Reengineering: Business Change of Mythical Porportions?", *Management Information Systems Quarterly*, July issue, 1994, pp. 121-127.

8. [Imai, 1992] Takuji Imai, *"The Next-Generation User Interface: From GUI to Multimodal"*, Nikkei Electronics Asia, 1992.

9. [Informix, 1994] Informix Press (1994). Informix 4GL: Reference, 6ª edição. Califórnia,

10. [Informix, 1999] Informix Press (1999). Informix Dynamic 4GL: The future of 4GL. Califórnia, Informix Press, USA.

11. [Informix, 2000] Informix Press (2000). "Informix Dynamic 4GL, Version 3.0 For Unix, Linux and Windows NT", Technical Brief, 3ª edição. Califórnia, Informix Press, USA.

12. [Lancaster, 1998] Lancaster University Press (1998). "Assessment of target system- elaborate evolution strategy", Esprit Project .Bailrigg, Lancaster University press, UK.

13. [Razak, 2001] Razak, Mohamed S. A.; Migração para GUI usando INFORMIX DYNAMIC 4GL Tese de Licenciatura. Pp 19 – 39. Maputo, Universidade Eduardo Mondlane.

14. [Renaissance, 1998a] Renaissance Web (1998). "The Renaissance of Legacy Systems", Esprit Project, Roma, Renaissance Web, Itália.

15. [Renaissance, 1998b] Renaissance Web (1998). Client/Server MIgration, 3 a edição, 110 pp., Renaissance Web, França.

16. [Renaissance, 1998c]Renaissance Web (1998). Modelling for System Evolution, 3a edição, 180 pp., Renaissance Web. Noruega.

17. [Ribeiro e Alexandre, 2004] Ribeiro, J. V. B. ; Alexandre J. P. I. N. (2004), *Four J's Business Development Language* Manual de Instalação/Utilizador, SINFIC SA, Portugal./

18. [SALEMI,1993] Salemi, Joe. "Base de Dados Cliente/Servidor" . IBPI Press, 1993.

19. [TONORIO, 2004] Jose Nelson Barbosa Tonorio. Um estudo sobre a utilização do Sistema de Informações nas pequenas empresas de confecção do Recife. Dissertação Submetida ao Colegiado do Programa Multiinstitucional e Inter-Regional de Pós-Graduação em Ciências Contábeis . UnB/UFPB/UFPE/UFRN. Recife. Agosto de 2004. http://www.unb.br/face/Disserta_Jose_Nelson_Tenorio.pdf

20. [URL-1] http://www.ptsi.pt/images/114/AReengenhariaDeSI.doc, CARLOS MIGUEL STOFFEL LEMOS, ABRIL 2004.

21. [URL-2] http://www.ppgia.pucpr.br/~maziero/ensino/sei/unix-ambiente.html

22. [URL-3]http://www.opus-software.com.br/produtos_sw_projects_conversao_informix_4gl_para_java.htm,OPUS SOFTWARE, Fevereiro/2004.

23. [URL-4] http://new.4js.com/fourjs/site/html/download_01.htm.

24. [URL-5] http://www.spi.pt/documents/books/inovint/gi/acesso_ao_conteudo_integral/capitulos/3.2/cap_apresentacao.htm.

25. [URL-6] http://www.unlu.edu.ar/~tyr/tyr/TYR-interior/Fundamentos_da_%20Arquitetura_Cliente-Servidor.pdf

26. [URL-7] Dalberto Adulis; Melhorando a Gestão Através do *Benchmarkin;* Tema do mês de março de 2002. http://www.rits.org.br/gestao_teste/ge_testes/ge_tmes_marco2002.cfm; Junho de 2005.

27. SPENDOLINI, M. J. (1992). Benchmarking, Tr. Kátia Aparecida Roque, São Paulo: Makron Books.

28. Amaral, W. H. "Arquitetura Cliente/Servidor Orientada a Objeto" Tese de Mestrado, IME, 1993.

29. Andrews, G. R. "Concurrent Programming: Principles and Practice" Benjamin/Cummings, Redwood City, CA, 1991.

30. Bancilhon, F. et al "Building an Object-Oriented Database System: The Story of O2", Morgan Kaufmann, 1992.

31. Bezerra, Eduardo (2002). Princípios de Análise de Projecto de Sistemas com UML. Rio de Janeiro: Campus.

32. Booch, Grady; Rumbaugh, James; Jacobson, Ivar (2000). UML, Guia do Usuário. Rio de Janeiro: Campus.

33. Botelho, Tomás de Aquino Tinoco, "Análise de Desempenho da Arquitectura Cliente/Servidor Orientada a Objecto", Tese de Mestrado, IME, Dezembro/1995.

34. Comer, Douglas E. & STEVENS, David L. "Internetworking With TCP/IP Vol.III: Client/Server Programming and Application (Socket Version)". Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

35. Custer, Helen "Windows NT", Microsoft Press, Makron Books, São Paulo, 1993.

36. Davis, Ralph "Windows NT Networking Programming" Addison Wesley, 1994

37. Deschamps, Eduardo R.P. "Sistema Gerenciador de Objetos em uma Arquitectura Cliente/Servidor", Tese de Mestrado, IME, Abril/1997.

38. Dumas, Arthur "Programando WinSock", Axcel Books, Rio de Janeiro, 1995.

39. Esselina Macome (Outubro-Novembro 1995). Introdução á Metodologia de Investigação. UEM, Faculdade de Ciências, (DMI).

40. EXI- Engenharia e Comercialização de Sistemas Informáticos. Contabilidade Pública, Manual de Utilizador, Versão 4.0; 2001.

41. http://216.239.37.104/search?q=cache:MjOJeBg9bhoJ:www.utrafe.gov.mz/Publicacoes/Modelo_de_gestao_da_UTRAFE.pdf+SISTAFE&hl=en&ie=UTF-8, ULTRAFE, Modelo de Gestão do SISTAFE, Março de 2004.

42. http://216.239.53.104/search?q=cache:57pTXG01lusJ:www.dcc.ufmg.br/~clarindo/pesquisa/gestus/artigousabmaturidade.pdf+qualidade+agradavel+de+interface&hl=en&ie=UTF-8, Professor Clarindo Isaías P. S. Pádua / Aloísio Antônio Ribeiro Júnior (mestrando), Departamento de Ciência da Computação da UFMG, Março de 2004.

43. http://geocities.yahoo.com.br/ergodesign_usabilidade/usabilid.htm, Usabilidade, ABERGO-Grupo Técnico de Ergodesign e Usabilidade de produtos de Informação e da Interacção humano-Computador, Março de 2004.

44. http://geocities.yahoo.com.br/interaface/interface.htm, INTERFACE, Março de 2004.

45. http://www.cbsconsulting.hpg.ig.com.br/o%20que%20e%20erp.htm, *Cid Barros da Silveira Netto*,Maio 2004.

46. http://www.eps.ufsc.br/disserta97/heemann/cap1.htm, avaliação de base de dados, dentro de um enfoque ergonómico, Capítulos I,II,III,IV,V,VI,VII - Vivian Heemann, Fevereiro/2004.

47. http://www.utrafe.gov.mz/e-sistafe%20english.htm, Ministério do Plano e Finanças, ULTRAFE, webmaster.utrafe@tvcabo.co.mz, Março de 2004.

48. Informix (Dezembro 1989). "INFORMIX-4GL USER GUIDE", Versão 4.0. Informix Software, Inc.

49. Informix (March,1990). "INFORMIX-4GL Reference Manual: Application Development Language for UNIX Operating System", Vol. 1, Versão 4.0. Informix Press, USA.

50. Informix (Setembro,1993). "INFORMIX-4GL For Windows: Development Tools", Versão Informix Press, USA.

51. Lancaster University Press (1996). "Evolution Planning", Esprit Project .Bailrigg, Lancaster University press, UK.

52. Lefebvre, Alain "L'Architecture Client-Serveur", Armand Colin, 2e édition, 1994.

53. Martin, James "Princípios de Análise e Projeto baseados em Objectos" Ed. Campus, Rio de Janeiro, 1994.

54. Navathe, Shamkant B. & Elmasri, Ramez "Fundamentals of Database Systems" 2nd Ed., Benjamin Cummings, CA, 1994.

55. Pham, Thuam Q. & Garg, Pankaj K. "Multithreaded Programming with Windows NT" Prentice Hall, 1996.

56. Renaud, P. E. "Introdução aos Sistemas Cliente/Servidor" IBPI Press, RJ 1993.

57. Roberts, Dave "Developing for the Internet with Winsock" Coriolis Group Books, 1995.

58. Tanenbaum, Andrew S. "Computer Networks" Prentice Hall, Third Edition, 1996.

59. Tanenbaum, Andrew S. "Distributed Operating Systems" Prentice Hall, 1995.

60. Tarouco, L. M. R. "Redes de Computadores". McGraw-Hill ,SP 1986.

10. ANEXOS

Anexo I

DEFINIÇÃO DE TERMOS

1.Sistemas Antigos

Os sistemas antigos (LS - "*Legacy Systems*") são normalmente compostos por diversos programas que partilham os mesmos dados (fig. 1) usando ficheiros e não sistemas de gestão de bases de dados (DBMS).

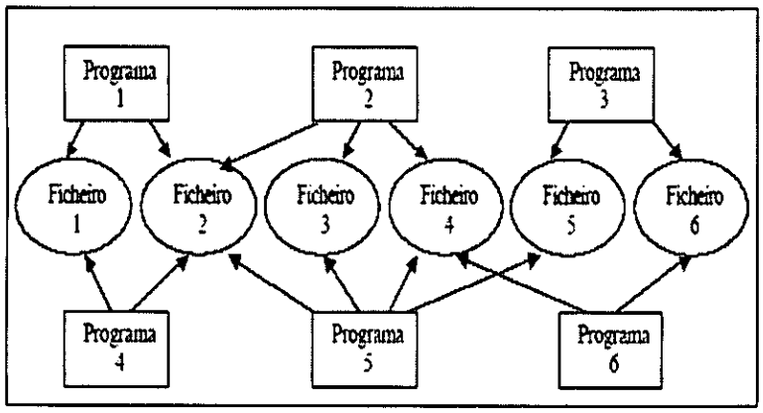


Fig. 1: Estrutura de um LS (*Legacy System*).

Os dados são geralmente duplicados devido à sua forma de representação em diversos ficheiros. Ao contrário dos DBMS, o armazenamento de dados em ficheiros não isola a estrutura de dados, uma vez que os programas aplicacionais que os partilham é que estão integrados com as estruturas de dados. Quaisquer modificações às estruturas de ficheiros podem ter repercussões por todo sistema [Renaissance, 1998b]. Esta situação torna-se complexa e difícil de gerir quando se pretende manter a consistência da duplicação dos dados em organizações com muitos sistemas que partilham dados usando ficheiros.

Muitos destes sistemas foram desenvolvidos antes da introdução das técnicas da programação estruturada, não permitindo a sua escalabilidade [Renaissance, 1998b].

2.Sistemas evolutivos

Sistemas evolutivos é um termo conceptual que descreve sistemas que estão capacitados para aderir às mudanças ao longo do seu ciclo de vida. Estes sistemas são baseados numa premissa sensível ao desenho de sistemas explicitamente direccionados para a evolução, uma vez que as mudanças são inevitáveis [Renaissance, 1998b].

A filosofia dos sistemas evolutivos é que nunca terminam, isto é, o sistema é inicialmente desenvolvido com uma capacidade e vai evoluindo ao longo do tempo, tornando-se mais capacitado. Esta filosofia quebra o modelo tradicional de desenvolvimento–manutenção, iniciando o novo modelo baseado na evolução contínua do sistema [Renaissance, 1998b].

3.Plataformas: *Mainframe* e Cliente/Servidor

Um *mainframe* é um computador de alto nível projectado para as tarefas computacionais mais intensas [Da Luz, Chaves e Nunes. 2001]. Os *mainframes* costumam ser compartilhados por diversos usuários conectados a eles através de terminais. Normalmente, são as grandes organizações que utilizam este tipo de plataforma no seu processamento de dados. Geralmente, executam-se no *mainframe* aplicações que atendem a toda a empresa, os chamados sistemas corporativos. Esses sistemas permitem o compartilhamento de dados com várias áreas da empresa de forma rápida e segura.

Já a arquitectura cliente/servidor é uma forma de estruturação das redes locais que utiliza o princípio da "distribuição da inteligência", tratando tanto o servidor quanto às estações de trabalho como equipamentos inteligentes e programáveis, permitindo a exploração plena de toda a capacidade instalada da rede[Da Luz, Chaves e Nunes. 2001]. Isto é feito através da divisão do processamento de uma aplicação entre dois componentes distintos: um cliente, no primeiro plano (*front-end*), e um servidor na retaguarda (*back-end*).

4.Bases de Dados: Hierárquicos, Relacionais e Orientados a Objecto.

Em termos gerais, pode-se definir Bases de Dados como qualquer conjunto de dados, mais especificamente, um arquivo ou tabela formado por uma série de registos, cada qual subdividido em campos ou colunas de tipo determinado, juntamente com um elenco de operações que facilitam a pesquisa, a classificação, a reorganização e outras actividades semelhantes [Da Luz, Chaves e Nunes. 2001].

Podemos dividir as Bases de Dados nas seguintes categorias: hierárquicos, redes, relacionais e orientados a objectos. Um Base de Dados hierárquico agrupa seus registos de modo que suas relações formam uma estrutura em forma de árvore. Bancos em rede é um tipo de Base de Dados onde os registos podem ser interligados em mais de uma maneira. Já as Bases de Dados relacionais armazenam as informações em tabelas – linhas e colunas de dados – e realizam pesquisas usando dados das colunas especificadas de uma das tabelas para encontrar dados complementares em outra tabela. A maioria dos *softwares* para micro-computadores possui características relacionais. Já as Bases de Dados orientados a objectos, têm como maior facilidade à armazenagem e a pesquisa não

só de letras e números, mas também imagens, vídeo, áudio, informações geoespaciais entre outros [Da Luz, Chaves e Nunes. 2001].

5.Interfaces: Caracter, Menus e Gráficas.

As interfaces com os usuários são formadas pelo projecto gráfico, pelos comandos, *prompts* e outros recursos que permitem ao usuário interagir com os programas. Existem três tipos básicos de interface com o usuário que são sempre mutuamente exclusivas: a interface de linha de comando, a baseada em menus e a gráfica:

- A interface de caracteres (linha de comando) é representada por um *prompt* que responde aos comandos digitados pelo usuário [Da Luz, Chaves e Nunes. 2001]. Como exemplo podemos citar o MS-DOS, UNIX, LINUX entre outros.
- Já as interfaces baseadas em menus, como as utilizadas nos terminais dos *mainframe's*, oferecem ao usuário a possibilidade de seleccionar comandos digitando uma letra ou uma tecla de direcção [Da Luz, Chaves e Nunes. 2001].
- Por fim, a interface gráfica tem como principal característica a utilização de um ambiente de janelas, colocando diante do usuário uma representação visual de alguma metáfora como a "mesa de trabalho" (*desktop*) e permite que o usuário controle, além das opções do menu, o tamanho, o leiaute e o conteúdo de uma ou mais janelas, ou áreas de trabalho no ecrã [Da Luz, Chaves e Nunes. 2001].

6. Benchmarking.

O *Benchmarking*[1] é uma metodologia utilizada por organizações para aperfeiçoar sua gestão mediante a realização sistemática de levantamentos e análises de práticas, processos, produtos e serviços prestados por outras organizações. O processo de benchmarking gera informações importantes para que as organizações conheçam diferentes formas de lidar com situações e problemas semelhantes e, desta forma, contribui para que as mesmas possam aperfeiçoar seus próprios processos de trabalho [URL-7].

[1]A palavra *benchmarking*, originária do idioma inglês, não tem tradução directa para o português. A definição mais usual para o conceito de benchmarking vem de Spendolini: Benchmarking é um processo contínuo e sistemático para avaliar produtos e processos de trabalho de organizações que são reconhecidas como representantes das melhores práticas, com a finalidade de melhoria organizacional (Spendolini, 1992).

Anexo II

Tabelas de comparação de extensões utilizadas pela IBM Informix e pela Four J's

Tabela 1: Compilação em P-Code [Ribeiro e Alexandre, 2004]

Fabricante	Four J's	IBM Informix
Produto	Four J's BDL	Informix 4GL-RDS
Ferramenta de compilação de ficheiros source .4gl	fgl2p	fglpc
Ferramenta de compilação de ficheiros source .per	fglform	Form4gl
Ferramenta de compilação de ficheiros source de mensagens .msg	fglmkmsg	mkmessage
Ferramenta para construção de bibliotecas	fgl2p	N/D
Debugger incorporado	Sim	Produto adicional: IBM Informix Interactive Debugger
Ferramenta de linkagem	fgl2p	Cat
Extensão utilizada para módulos em P-Code	.42m	.4go
Extensão utilizado para programas em P-Code	.42r	.4gi
Extensão utilizada para forms compilados	.42f	.frm
Extensão utilizada para bibliotecas	.42x	N/D
Extensão utilizada para mensagens compiladas	.iem	.iem
Linker inteligente?	Sim	Não

Carregamento dinâmico de módulos ?	Sim	Não
Os módulos têm de estar presentes para a execução dos programas ?	Sim	Não
Execução em modo gráfico	Sim	Não
Suporte para plataforma Windows	Sim	Não
Execução via Browser	Sim	Não
Controle de licenças	Sim	Não

Tabela 2: Compilação em C-Code [Ribeiro e Alexandre, 2004]

Fabricante	Four J's	IBM Informix
Produto	Four J's BDL	Informix 4GL
Ferramenta de compilação de ficheiros source .4gl	fgl2c	C4gl
Ferramenta de compilação de ficheiros source .per	fglform	Form4gl
Ferramenta de compilação de ficheiros source de mensagens .msg	fglmkmsg	mkmessage
Ferramenta para construção de bibliotecas	ar42o	ar
Ferramenta de linkagem	fgl2c	C4gl
Extensão utilizada para módulos objectos em C-Code	.42o	.o
Extensão utilizado para Programas em C-Code	.42e	.4ge

Extensão utilizada para forms compilados	.42f	.frm
Extensão utilizada para bibliotecas	.a	.a
Extensão utilizada para mensagens compiladas	.iem	.iem
Linker inteligente?	Sim	Sim?
Carregamento dinâmico de módulos ?	Não	Não
Os módulos têm de estar presentes para a execução dos programas ?	Não	Não
Execução em modo gráfico	Sim	Não
Suporte para plataforma Windows	Sim	Não
Execução via Browser	Sim (?)	Não
Controle de licenças	Sim	Não

Tabela 3: Estrutura da directoria de instalação do Four J's *Business Development Support* (BDS) ($FGLDIR) [Ribeiro e Alexandre, 2004]

Path	Decrial
bin	Ficheiros executáveis e bibliotecas partilhadas *(shared libraries)*
Bmp	Desenhos utilizados nos exemplos do Front End X11.
Defaults	Program específicos da configuração do FGLPROFILE.
Demo	Programas de demonstração do Four J's BDL.
Desi	Programa de configuração do Front End X11
Etc	Ficheiros de confiuração e recursos do cliente gráfico.
Etc/ger	Filtro de caracteres para o alfabeto alemão.
Etc/iso	Filter for para o conjunto de caracteres ISO.
Include	Ficheiros de *include.*
include/f2c	Ficheiros de *include* para os interfaces de C.
include/sql	Ficheiros *header* para o driver SQL.
Lib	Ficheiros 4GL e bibliotecas DVM (Dynamic Virtual Machine) para linkar um runner.
Lock	Ficheiros de dados de controle de licenças *(não apagar!).*
Msg	Ficheiros compilados de mensagens.
odi	Directoria de ferramentas do Open Database Interface.
Release	Documentação mais recente (release notes).
Src	Ficheiros source de ferramentas e bibliotecas 4GL.

Toolbars	Icons para a Toolbar.

Tabela 4: Sistemas operativos suportados [Ribeiro e Alexandre, 2004]

Sistema Operativo	ID	Processador	Arquitectura	Versão
Compaq Tru64	OSF	Alpha	64 bit	4.0
DG/UX	DGX	X86	32 bit	R4.20
Dynix/PTX	DNX	X86	32 bit	4.4.2
HP/UX	HPX	Risc	32 bit	11
HP/UX 64	H64	Risc	64 bit	11
IBM AIX	AIX	Risc	32 bit	4.3
IBM AIX 64	A64	Risc	64 bit	4.3
Linux	LNX	X86	32 bit	libc 2.2
Reliant UNIX	SNX	X86	32 bit	5.43
SCO Open Server	**SCO**	**X86**	**32 bit**	**5.0.x**
SCO Unixware	UXW	X86	32 bit	7.1
SGI IRIX	IRX	Risc	32 bits (N32)	6.5
SGI IRIX 64	I64	Risc	64 bit	6.5
SUN Solaris	SLI	X86	32 bit	2.6
SUN Solaris	SLS	Sparc	32 bit	2.6
MS Windows	**WNT**	**X86**	**32 bit**	**NT4/SP5 & 2000**

EXTENSÕES DE LINGUAGEM FOURJS BDL

Inicialização de variáveis

O Four J's BDL possibilita 3 tipos de variáveis [Ribeiro e Alexandre, 2004]:

1.Variáveis Globais

2.Variáveis locais ao módulo

3. Variáveis locais (variáveis da função ou do *report*)

Todas a variáveis são inicializadas automaticamente, a quando da sua declaração, de acordo com o seu tipo de dados. A maioria são inicializadas com a constante **NULL**, mas existe certo tipo de dados numéricos que são inicializados com o valor zero (Integer, Smallint, Float e Smallfloat) [Ribeiro e Alexandre, 2004].

Variáveis de ambiente

Descrição das Variáveis de ambiente e respectivos valores possíveis, incluindo exemplos de afectação em ambientes UNIX e Windows.

Algumas das variáveis de ambiente estão disponíveis apenas em sistemas UNIX.

***NOTA**: em plataformas Windows o interpretador de comandos (cmd.exe para*

plataformas NT/2000/XP/2003 ou command.com para Windows 95/98/Me) interpreta as aspas (") como um caractere normal e não como delimitador de strings como acontece no UNIX! Como tal, se afectarmos uma variável de ambiente com o comando SET VAR="xxx", temos 5 (cinco) caracteres na variável de ambiente.

Variáveis de ambiente comuns

Variáveis de ambiente comuns utilizada pela Four J's.

PATH

Esta variável de ambiente contém o caminho para os programas executáveis.

Adicionar ao *path* a directoria de binários da Four J's (FGLDIR/bin). Os valores possíveis são:

Em UNIX, as directorias são separadas por dois pontos (:), em Windows as directorias são separadas por ponto e virgula (;).

Exemplos:

Korn Shell $ export PATH=$PATH:$FGLDIR/bin C Shell % set path = ($FGLDIR/bin $path) MS-DOS C:\>set PATH=%FGLDIR%\bin;%PATH%

CC

Esta variável de ambiente contem o nome do compilador de C por omissão utilizado para a compilação de ficheiros em linguagem C.

Utilizada apenas em plataformas UNIX. Os valores possíveis são:

O nome do compilador de C instalado na maquina (normalmente "cc" ou "gcc").

Exemplos:

Korn shell $ export CC="cc" C Shell $ setenv CC "cc"

GCC

Esta variável de ambiente contem o nome do compilador de C da GNU.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

O nome do compilador de C da GNU instalado na maquina (normalmente 'gcc').

Exemplos:

Korn shell $ export GCC="gcc" C Shell $ setenv GCC "gcc"

GCCDIR

Esta variável de ambiente contem o caminho para a directoria de instalação do compilador GCC. *Utilizada apenas em plataformas UNIX.*

Os valores possíveis são:

A directoria de instalação do compilador GCC.

Exemplos:

Korn shell $ export GCCDIR="/usr/local/gcc-2.95" C Shell $ setenv GCC "/usr/local/gcc-2.95"

GCC_EXEC_PREFIX

Esta variável de ambiente contem o caminho para a execução dos binários do

compilador GCC.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

A directoria de execução dos binários do compilador GCC.

Exemplos:

Korn shell $ export GCC_EXEC_PREFIX="/usr/local/gcc- 2.95" C Shell $ setenv GCC_EXEC_PREFIX"/usr/local/gcc- 2.95"

TCLDIR

Esta variável de ambiente é utilizada apenas com o pacote Tcl/Tk incluído no

FourJs BDS. Esta variável de ambiente especifica o caminho completo para a

directoria de instalação do Tcl/Tk

Utilizada apenas em plataformas UNIX.

***NOTA**: Incluir a directoria $TCLDIR/bin no PATH.*

Os valores possíveis são:

O caminho completo para a directoria de instalação do Tcl/Tk.

Exemplos:

Korn shell $ export TCLDIR="/opt/4js/tcltk" C Shell $ setenv TCLDIR "/opt/4js/tcltk"

TK_LIBRARY

Esta variável de ambiente especifica o caminho completo para as bibliotecas de

TK.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

O caminho completo para a biblioteca de TK.

Exemplos:

Korn shell $ export TCLDIR="/opt/4js/tcltk/lib/tk" C Shell $ setenv TCLDIR "/opt/4js/tcltk/lib/tk"

TCL_LIBRARY

Esta variável de ambiente especifica o caminho completo para as bibliotecas de

TCL.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

O caminho completo para a biblioteca de TCL.

Exemplos:

Korn shell $ export TCLDIR="/opt/4js/tcltk/lib/tcl" C Shell $ setenv TCLDIR "/opt/4js/tcltk/lib/tcl"

Variáveis de ambiente especificas do FourJs BDS

Descrição das Variáveis de ambiente especificas do FourJs BDS.

FGLDIR

Esta variável de ambiente define o caminho para a directoria de instalação do

produto. Esta variável é obrigatória para a utilização dos vários componentes

dos produtos da FourJs.

Os valores possíveis são:

String com a directoria de instalação do FourJs BDS.

O valor por omissão é:

/usr/fgl2c em UNIX, **C:\USR\FGL2C** em plataforma Windows.

Exemplos

Korn shell $ export FGLDIR="/opt/4js/bds/compiler" C Shell % setenv TCLDIR "/opt/4js/bds/compiler" MS-DOS C:\>set FGLDIR=C:\USR\ACCT\RT4JS

FGLRUN

Esta variável de ambiente define o nome do runner de P-Code especifico criado

com a ferramenta fglmkrun. É utilizada por vários componentes, como por exemplo: fgl2p

Os valores possíveis são:

O nome do runner especifico criado com fglmkrun.

O valor por omissão é:

Não existe valor por omissão (nas ferramentas o valor por omissão é "fglrun").

Exemplos

Korn shell $ export FGLRUN="dir_da_aplicacao/bin/myrun"

C Shell % setenv FGLRUN "dir_da_aplicacao/bin/myrun"

MS-DOS C:\>set FGLRUN=dir_da_aplicacao\bin\myrun

FGLLDPATH

Esta variável de ambiente fornece ao runner de P-Code o caminho correcto de

procura dos módulos objectos de P-Code (os módulos ".42m"), que são dinamicamente linkados num programa executável de P-Code.

Os valores possíveis são:

O caminho para a directoria dos módulos de P-Code.

O valor por omissão é:

A directoria actual.

Exemplos

Korn shell $ export FGLLDPATH="dir_da_aplicacao/bin" C Shell % setenv FGLLDPATH "dir_da_aplicacao/bin" MS-DOS C:\>set FGLLDPATH=dir_da_aplicacao\bin

FGLSQLDEBUG

Esta variável de ambiente define o nível de rastreamento (trace) das instruções

SQL. A informação de depuramento (debug) é mostrada no output do *standard*

error (normalmente o ecrã do terminal).

Os valores possíveis são:

0, 1, ... N (zero desliga o rastreamento de SQL).

O valor por omissão é:

0 (zero)

Exemplos

Korn shell $ export FGLSQLDEBUG=3 C Shell % setenv FGLSQLDEBUG 1 MS-DOS C:\>set FGLSQLDEBUG=0

FGLCC

Esta variável de ambiente define o compilador de C por omissão utilizado na

criação de runners de P-Code ou na compilação de programas em C-Code.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

O nome do compilador de C instalado na máquina (normalmente "cc" ou "gcc").

Exemplos

Korn shell $ export FGLCC="gcc" C Shell % setenv FGLCC "gcc" FGLLIBSQL

Esta variável de ambiente define o caminho completo para as bibliotecas SQL,

utilizadas na linkagem no runner especifico de P-Code de funções de C de interface com o servidor de base de dados.

Utilizada apenas em plataformas UNIX.

Os valores possíveis são:

O caminho completo para as bibliotecas SQL.

O valor por omissão é:

$INFORMIX/lib/libfesql.a

Exemplos

Korn shell $ export FGLLIBSQL="$INFORMIXDIR/lib/libfesql.a" C Shell % setenv FGLLIBSQL "$INFORMIXDIR/lib/libfesql.a"

FGLLIBSYS

Esta variável de ambiente define a lista completa de bibliotecas de sistema e opções utilizadas na compilação de um runner especifico de P-Code ou programas de C-Code.

Os valores possíveis são:

O caminho completo para as bibliotecas SQL.

O valor por omissão é:

Dependente do sistema operativo.

Exemplos

Korn shell $ export FGLLIBSYS="-lm -lsocket"

C Shell % setenv FGLLIBSYS "-lm -lsocket"

FGLGUI

Esta variável de ambiente define se as aplicações iram ser executadas com um

interface baseado em caracteres (UIBC) ou com um interface gráfico (GUI)

Os valores possíveis são:

0 (zero) : execução de aplicações em modo de caracteres (UIBC);

1 (um) : execução de aplicações em modo gráfico (GUI);

2 (dois) : HTML.

O valor por omissão é:

0 em UNIX, 1 em plataformas Windows.

Exemplos

Korn shell $ export FGLGUI=1 C Shell % setenv FGLGUI 1 MS-DOS C:\>set FGLGUI=1

FGLDBPATH

Esta variável de ambiente define o caminho para os ficheiros de esquema (gerados pelos componentes "fglschema" ou "fgldbsch") das várias bases de dados utilizadas, separadas por dois pontos (:) em sistemas UNIX ou por ponto e virgula em plataformas Windows. O compilador não acede as bases de dados mas sim aos esquemas específicos gerados pelos componentes "fglschema" ou

"fgldbsch".

Os valores possíveis são:

Qualquer string

O valor por omissão é:

A directoria actual.

Exemplos

Korn shell $ export FGLDBPATH="dir_da_aplicacao/schema" C Shell % setenv FGLDBPATH "dir_da_aplicacao/schema" MS-DOS C:\>set FGLDBPATH=dir_da_aplicacao\schema

FGLSOURCEPATH

Esta variável de ambiente define o caminho para os ficheiros sources BDL (ou seja, para os ficheiros .4gl) utilizados pelo debugger.

Os valores possíveis são:

Qualquer string

O valor por omissão é:

A directoria actual.

Exemplos

Korn shell $ export FGLDBPATH="dir_da_aplicacao/dir1:dir_da_aplicacao_dir2"

C Shell % setenv FGLDBPATH "dir_da_aplicacao/dir1:dir_da_aplicacao_dir2"

MS-DOS C:\>set FGLDBPATH=dir_da_aplicacao\dir1;dir_da_aplicacao_dir2

NOTA: Ter também em atenção as seguintes variáveis de ambiente:

Variáveis de ambiente genéricas:

LD_LIBRARY_PATH (apenas para alguns sistemas UNIX)

SHLIB_PATH (apenas para alguns sistemas UNIX)

LIBPATH (apenas para alguns sistemas UNIX)

CC (apenas para sistemas UNIX)

TERMCAP (apenas para sistemas UNIX)

LANG (apenas para sistemas UNIX)

LC_ALL (apenas para sistemas UNIX)

LC_COLLATE (apenas para sistemas UNIX)

LC_CTYPE (apenas para sistemas UNIX)

LC_MESSAGES (apenas para sistemas UNIX)

LC_MONETARY (apenas para sistemas UNIX)

LC_NUMERIC (apenas para sistemas UNIX)

LC_TIME (apenas para sistemas UNIX)

LC_TYPE (apenas para sistemas UNIX)

TEMP TMP TMP_DIR

Variáveis de ambiente da FourJs:

FGLCFLAGS FGLDBS FGLLFLAGS

FGLPROFILE

FGLPS

FGLSERVER

FGLSHELL

Variáveis de ambiente da IBM Informix:

INFORMIXDIR

INFORMIXSERVER INFORMIXSQLHOSTS INFORMIXHOST (apenas para plataformas Windows)

INFORMIXSERVICE (apenas para plataformas Windows)

INFORMIXPROTOCOL (apenas para plataformas Windows)

INFORMIXTERM INFORMIXC (apenas para sistemas UNIX)

ONCONFIG (apenas para IBM Informix Dynamic Server)

SQLEXEC

DBDELIMITER

DBMONEY

DBPATH

DBPRINT

DBDATE

DBTIME

DBCENTURY

DBEDITOR

DBTEMP

DBSPACETEMP

GL_DATE

GL_DATETIME

DB_LOCALE

SERVER_LOCALE

CLIENTE_LOCALE

Anexo III

LISTAGENS EXTRAÍDAS PARA EFEITOS DE TESTES

Tabela 5: Dados de requisições a cabimentar

Requisição nº	1	2	3
Codigo de Etidade	69	69	141
Classificação Orçamental	12 2001	12 2001	12 1005
Classificação Funcional	1010	1010	1010
Valor Requisitado	19,531,598.00	20,580,258	351,000.00
Data de Emissão de Requisição	09/09/2004	09/09/2004	09/09/2004

Tabela 6: Listagem extraida por consulta a bases de dados

UIBC antigo	UIBC novo	GUI novo
[redacted]	data 09/09/2004 descricao CABIMENTACAO tip_mov C quantia 20580258.00 quant_aux 224000000.00 r_tip REQ r_num 2 tipo_imputacao D class_organica 12 class_economica 2001 class_funcional 1010 l_tip l_num centid 69 1 row(s) retrieved.	data 09/09/2004 descricao CABIMENTACAO tip_mov C quantia 351000.00 quant_aux 120000000.00 r_tip REQ r_num 3 tipo_imputacao D class_organica 12 class_economica 1005 class_funcional 1010 l_tip l_num centid 141 1 row(s) retrieved.

Anexo IV

EXEMPLOS DE FICHEIROS EM I4GL

1. Definição de variáveis globais

Fich: **cpglobals.4gl**

```
DATABASE "publica"

GLOBALS

DEFINE  pr_ccent RECORD LIKEccent.*,     pr_ccotRECORD LIKEccot.*,     pr_desp0RECORD LIKEdesp0.*,
pr_desp1RECORD LIKEdesp1.*,   pr_desp2RECORD LIKEdesp2.*,

  pr_desphRECORD LIKEdesph.*,

.....

 class_otLIKE ot0.class,  numaltrevINTEGER,  datahistDATE,  opcao_saidaCHAR (1),  opcao_dotCHAR (1),

 init_docINTEGER,  opcao_dataDATE,  data_trab  DATE,  camaraCHAR (50),  camara_sSMALLINT,
wbufCHAR {(300)}(1000),  qbufCHAR {(500)}(1000),

 maisCHAR (1),

.....

sys_rec RECORD
        nome      CHAR (30),
        formato   SMALLINT,
        sigla     CHAR (5),
        nvl       SMALLINT,
        area      CHAR(10),
        nome_logico CHAR (10),
        comp_on,comp_off,          # impressao em compressed.
        under_on,under_off,        #   "    " underline.
        bold_on,bold_off,          #   "    " bold.
        dwide_on,dwide_off CHAR (20) #   "    " double-wide.
      END RECORD,

 DEFINE Ar_Ordem, Ar_ordemFi ARRAY[20] OF RECORD

 label   CHAR(20), posicao INTEGER, direccao CHAR(4), campo   CHAR(100)    END RECORD,  stOrdem
CHAR(200),ImporOrdem CHAR(1)

END GLOBALS
```

2. Definição do ficheiro antigo - módulo de contabilidade

Fich: **contbmain.4gl**

```
# (C) Copyrigth EXI, S.A.
# DATAS: 04-02-1992--- ?-?-1992
# FUNCOES:
# DESCRICAO:
#   Modulo principal do programa impress.4gi.
GLOBALS "cpglobals.4gl"
MAIN
 DEFINE comando char(100)
 LET sys_versao = "UNIX"
IF sys_versao = "DOS" THEN  LET comando = "\tmp\pub.err"  ELSE       LET comando = "/tmp/pub.err"
END IF
 LET hoje = TODAY  CALL startlog(comando)  WHENEVER ERROR CALL SysBDErr
 DEFER INTERRUPT   SET LOCK MODE TO WAIT  OPTIONS    FORM LINE 1,   PROMPT LINE LAST
 let tx = 1
 IF NOT SysExistemImpressoras () THEN    CALL SysBottomMsg ("AINDA NAO FORAM CRIADAS
IMPRESSORAS")
 END IF
 CLEAR SCREEN
END MAIN
FUNCTION SysAccoes (accao)
DEFINE accao CHAR(40) DEFINE msg CHAR (80)
 CASE accao
.....
   WHEN "Gestao-de-Terminais"      CALL SysGereTerminais (2,2)
         WHEN "cpftrcp"            if v_tesou = "n" then            error "tesouraria nao esta autorizada"
                                exit case
                                      end if
         CALL cpftrcp("C")
 WHEN "cpftrct"  if v_tesou = "n" then  error "o pessoal na esta instalado"   exit case  end if  CALL cpftrct("C")
```

.....

WHEN "cpb_ddia"

CALL cpb_ddia()

.....

END CASE

END FUNCTION

3. Definição nova de módulo de contabilidade para efeito de lincagem

Fich: contabilidade.link.4js

Modulo da Biblio

BLI_DIR=/u/exilib/lib4gl

echo "A linkar o modulo de Contabilidade - contabi.42r"

$FGLDIR/bin/fgl2p -o contabi.42r \

$BLI_DIR/sysoper.42m \$BLI_DIR/sysimpr.42m \$BLI_DIR/systerm.42m \$BLI_DIR/sysform.42m \

.....

$BLI_DIR/sysmes.42m \$BLI_DIR/sysgui.42m \cpglobals.42m \contbmain.42m \

.....

cpfalocoiv.42m \msgmsg.42m \mapmsg.42m \relogio.42m \sysdivide.42m \cpfordem.42m \

echo "done."

4. Definição de funções ou subrotinas e variáveis locais

Fich: **cpfcaddr2.4gl**

GLOBALS "cpglobals.4gl"

FUNCTION cpb_ddia ()

DEFINE pr_ecran RECORD data LIKE requisicao.data, num_req LIKE requisicao.num_req,

centid LIKE requisicao.centid, estado Like requisicao.estado END RECORD,

pr_ecran2 ARRAY[30] OF RECORD num_reqint LIKE reqfundos.num_reqint,

class_funcional LIKE reqfundos.class_funcional, requisitado LIKE reqfundos.requisitado,

ref_ano LIKE reqfundos.ref_ano END RECORD,

f_entnome LIKE entid.nome, naoreqLIKE desp0.dotacao, f_descLIKE desp0.descricao, v_data,data1 DATE,

```
i, num,num, v_seg, v_numreq, dias_uteis, cont, mensalre,anualre,naoreq2, totreq,acttotreq,anttotreq,react, doact
LIKE reqfundos.requisitado, meses_serv SMALLINT, prob CHAR(1), v_che CHAR(1),
 v_che1,opcao CHAR(1), actual CHAR(4), filho CHAR(10), estado CHAR(15)
OPEN WINDOW wrequis AT 3,6 WITH FORM "f_reqfund" ATTRIBUTE (BORDER)
MENU "REQUISICAO"
 COMMAND "Pesquisar" "Pesquisa a tabela activa" CALL pesq_requisicao (v_data, v_numreq)
 COMMAND "Sair" "Sair da Opcao" EXIT MENU
END MENU
CLOSE WINDOW wrequis
END FUNCTION
.....
REPORT pprrequisicao (pr_ecran, p_nome,estado_aux)
DEFINE
 pr_ecran RECORD LIKE requisicao.*, p_nome LIKE entid.nome,
pr_ecran2 RECORD num_reqint LIKE reqfundos.num_reqint,
        class_funcional LIKE reqfundos.class_funcional, requisitado LIKE reqfundos.requisitado,
        ref_ano LIKE reqfundos.ref_ano END RECORD,
 estado_aux CHAR(15), f_desc LIKE desp0.descricao, totreq LIKE reqfundos.requisitado,
    v_che CHAR(1), vez,i SMALLINT
FORMAT
PAGE HEADER
 IF pageno = 1 THEN PRINT "/[" ELSE PRINT
 END IF
 PRINT COLUMN 1,"+----------------------------------------------------------
----------+"
 PRINT COLUMN 1,"| ",empresa_rec.designacao,column 67,"SIGLA: ",sys_tty CLIPPED
," ","|"
 PRINT COLUMN 1,"|",column 80,"|"
 PRINT COLUMN 1,"|          R E Q U I S I C A O  D E  F U N D O S          |"
 PRINT COLUMN 1,"|",column 80,"|"
```

```
 PRINT COLUMN 1,"| ","DATA: ",hoje,COLUMN 33,"HORA: ",time,COLUMN 71,"PAG: ",
     COLUMN 77,pageno USING "##"," |"
 PRINT COLUMN 1,"+----------------------------------------------------------
----------+"
 SKIP 2 LINES
FOR i = 1 TO 80  PRINT "-";END FOR
PRINT
 PRINT COLUMN 1,"DATA: ", COLUMN 8,pr_ecran.data, COLUMN 35,"ESTADO: ",
           estado_aux
PRINT COLUMN 1,"REQ. No: ",COLUMN 9,pr_ecran.num_req USING "<<<<<"
 PRINT COLUMN 1,"ENTIDADE: ",COLUMN 12,pr_ecran.centid USING "<<<<<",
    COLUMN 20,p_nome CLIPPED    SELECT descricao    INTO f_desc    FROM desp0    WHERE class_organica
= pr_ecran.class_organica AND         class_economica = pr_ecran.class_economica
 PRINT    COLUMN    1,"    CLASS.    ORCAMENTAL:    ",pr_ecran.class_organica    CLIPPED,"|",
pr_ecran.class_economica CLIPPED," ",f_desc  SKIP 1 LINES
FOR i = 1 TO 80  PRINT "-";END FOR
PRINT
 PRINT COLUMN 1, "REQUISICAO    CLASSIFICACAO        VALOR        REFERENTE"
 PRINT COLUMN 1, " INTERNA        FUNCIONAL        REQUISITADO        ANO"
FOR i = 1 TO 80  PRINT "-"; END FOR
PRINT
BEFORE GROUP OF pr_ecran.num_req   SKIP TO TOP OF PAGE
ON EVERY ROW   DECLARE cur CURSOR FOR    SELECT num_reqint,class_funcional,requisitado,ref_ano
    FROM reqfundos      WHERE num_req = pr_ecran.num_req      ORDER BY num_reqint,class_funcional
 LET totreq = 0   FOREACH cur INTO pr_ecran2.*     LET totreq = totreq + pr_ecran2.requisitado
   PRINT COLUMN 2, pr_ecran2.num_reqint USING "####",    COLUMN 20,pr_ecran2.class_funcional, COLUMN
27, pr_ecran2.requisitado USING "### ### ### ### ##&.&&", COLUMN 55,pr_ecran2.ref_ano
          USING "####"   END FOREACH
FOR i = 1 TO 80   PRINT "-";END FOR PRINT   PRINT COLUMN 1,"    TOTAL DA REQUISICAO ", COLUMN
27,totreq              USING "### ### ### ### ##&.&&"FOR i = 1 TO 80  PRINT "-";END FOR
PRINT  ON LAST ROW  PRINT "/["  END REPORT
```

```
.....

FUNCTION cabreq(v_data,v_numreq)

DEFINEv_data,data1  DATE, v_numreq  LIKE requisicao.num_req, cont, #v_numreq  SMALLINT,

 congel  DECIMAL(17,2),      anterior  DATE,      prob    CHAR(1),      v_che CHAR(1),b  decimal(17,2),

 pr_ecran    RECORD  LIKE requisicao.*,      doc_rowid1 INTEGER

SELECT *  INTO pr_ecran.*  FROM requisicao WHERE num_req = v_numreq LET prob = "n"

IF pr_ecran.estado = "R" THEN   ERROR "NAO PODE CABIMENTAR UMA REQUISICAO ANULADA"

 SLEEP 2

 LET prob = "s"  END IF

 IF (pr_ecran.estado = "C") AND (prob = "n")  THEN   ERROR "ESTA REQUISICAO JA FOI CABIMENTADA"

  SLEEP 2

  LET prob = "s"  END IF

 IF (pr_ecran.estado = "A") AND (prob = "n") THEN

    SELECT  COUNT  (*)          INTO cont          FROM  ddia          WHERE  tip_mov  =  "C"      AND  r_num  =
pr_ecran.num_req      IF (cont > 0)       THEN

        ERROR " JA EXISTE UMA CABIMENTACAO LANCADA COM ESTA REF/DOC "

        LET prob = "s"          sleep 5     END IFEND IF     IF (year(hoje) <> anocorrenteF())       THEN

       ERROR "NAO PODE CABIMENTAR POIS ANO DO SISTEMA DIFERENTE DO ANO CORRE
NTE"          LET prob = "s"          sleep 4     END IF#error "prob1 = ",prob#sleep 5IF prob = "n" THEN

.....

  ERROR " DOTACAO NAO CABIMENTADA DESTA RUBRICA = ",pr_ddia.quant_aux USING "--
----------.##"

WHEN 2

  LET pr_ddia.quant_aux = pr_desp0.dotacao  - congel - pr_desp0.cab1 - pr_desp0
.cab2 - pr_desp0.cab3 - pr_desp0.cab4 - pr_desp0.cab5 - pr_desp0.cab6 - pr_desp0
.cab7 - pr_desp0.cab8 - pr_desp0.cab9 - pr_desp0.cab10 - pr_desp0.cab11 - pr_des
p0.cab12 + pr_desp0.ref2

ERROR " DOTACAO NAO CABIMENTADA DESTA RUBRICA = ",pr_ddia.quant_aux USING "--
----------.##"

WHEN 3
```

```
  LET pr_ddia.quant_aux = pr_desp0.dotacao  - congel - pr_desp0.cab1 - pr_desp0
.cab2 - pr_desp0.cab3 - pr_desp0.cab4 - pr_desp0.cab5 - pr_desp0.cab6 - pr_desp0
.cab7 - pr_desp0.cab8 - pr_desp0.cab9 - pr_desp0.cab10 - pr_desp0.cab11 - pr_des
p0.cab12 + pr_desp0.ref3
  ERROR " DOTACAO NAO CABIMENTADA DESTA RUBRICA = ",pr_ddia.quant_aux USING "--
----------.##"
.....
WHEN 12
  LET pr_ddia.quant_aux = pr_desp0.dotacao  - congel - pr_desp0.cab1 - pr_desp0
.cab2 - pr_desp0.cab3 - pr_desp0.cab4 - pr_desp0.cab5 - pr_desp0.cab6 - pr_desp0
.cab7 - pr_desp0.cab8 - pr_desp0.cab9 - pr_desp0.cab10 - pr_desp0.cab11 - pr_des
p0.cab12 + pr_desp0.ref12
ERROR " DOTACAO NAO CABIMENTADA DESTA RUBRICA = ",pr_ddia.quant_aux USING "--
----------.##"

END CASE
####era este
############ERROR " DOTACAO NAO CABIMENTADA DESTA RUBRICA = ",pr_ddia.quant_aux
USING "----------.##"
SLEEP 2
#ERROR "antes do foreach "
#SLEEP 2
 FOREACH c_x INTO  pr_ddia.class_funcional,pr_ddia.quantia
 # ERROR " quant_aux ",pr_ddia.quant_aux," requisitado ",pr_ddia.quantia," rubr
ica ",pr_ddia.class_funcional
 # Sleep 5
  IF (pr_ddia.quantia > pr_ddia.quant_aux) THEN
     ERROR " A DOTACAO DISPONIVEL DESTA RUBRICA NAO E SUFICIENTE PARA ESTE CAB
IMENTO "
SLEEP 2
```

```
    ERROR " SERA NECESSARIO RE-ALOCAR FUNDOS A ESSA RUBRICA"
.....
RETURN v_che
END FUNCTION
```

5. Definição de formulários

Fich: **f_reqfund.per**

```
database publica
screen
{

---------------------------------------------------------------
          REQUISICAO DE FUNDOS EM [f000     ]
---------------------------------------------------------------
REQUISICAO No  [f001       ] ESTADO REQ.  [f012       ]
COD. ENTIDADE  [f002 ] [f003                    ]
---------------------------------------------------------------
CLASSIFICACAO ORCAMENTAL  [f004 |f005  ]
[f006                    ]
ADT/FMA [f013 ] Num [f014   ]
Dotacao ainda nao requisitada [d000       ]

REQUISICAO   CLASSIFICACAO        VALOR          REFERENTE
 INTERNA      FUNCIONAL         REQUISITADO       AO ANO
[f011   ]    [f007   ]      [f008       ]     [f009]
[f011   ]    [f007   ]      [f008       ]     [f009]
[f011   ]    [f007   ]      [f008       ]     [f009]
[f011   ]    [f007   ]      [f008       ]     [f009]
[f011   ]    [f007   ]      [f008       ]     [f009]
          TOTAL DO ORCAMENTO [d001
```

```
}
end
tables
requisicao
reqfundos
attributes
f000 = requisicao.data,reverse,autonext,default=today,format="dd/mm/yyyy";
f001 = requisicao.num_req,reverse,autonext;
f002 = requisicao.centid,reverse,autonext;
f003 = FORMONLY.f_entnome,reverse,autonext;
f004 = requisicao.class_organica,reverse,autonext;
f005 = requisicao.class_economica,reverse,autonext;
f006 = FORMONLY.f_desc,reverse,autonext;
f007 = reqfundos.class_funcional,reverse,autonext;
f008 = reqfundos.requisitado,reverse,autonext;
f009 = reqfundos.ref_ano,reverse;
f011 = reqfundos.num_reqint,reverse,autonext;
f012 = FORMONLY.estado,reverse,autonext;
f013 = FORMONLY.t_adiant,reverse,upshift;
f014 = FORMONLY.num_adiant,reverse;
d000 = FORMONLY.naoreq TYPE LIKE reqfundos:requisitado,reverse,autonext;
d001 = FORMONLY.totreq TYPE LIKE reqfundos.requisitado,reverse,autonext;
end
instructions
 delimiters ".."
 screen record sr_rf (requisicao.data,requisicao.num_req,requisicao.centid,requ
isicao.class_organica,requisicao.class_economica)
 screen record sr_reqfund[5] ( reqfundos.num_reqint,reqfundos.class_funcional,
                    reqfundos.requisitado,reqfundos.ref_ano)
.....
```

Fich: **f_real.per**

```
database FORMONLY
screen
{
----------------------------
 DESEJA FAZER A RE-ALOCACAO
     DE FUNDOS ?
----------------------------
   Sim / Nao / Aborta


            OPCAO : [a]
}
end
attributes
a = FORMONLY.v_che,autonext,reverse,upshift;
end
instructions
  delimiters " "
```

6. Definição do ficheiro antigo makefile I4GL

Ferramenta que ajuda o programador a manter, actualizar e regenera programas do computador, contem a sequência de entradas e especificações de dependências

Fich: **makefile**

```
C = c4gl
#P = /home1/publico/lib/funcrds/
#P = /u/exilib/lib_obj/
P = /u/exilib/lib_obj/
FLAGS = c
O = .o
EXECFLAG = -o
EXEC = .4ge
COMP = c4gl
NOVANO = cpglobals$(O) novamain$(O) cprdesp1$(O) cpfcrina$(O) cpfinimes$(O)\
 cpfradep1$(O) cpfrdep1$(O) cpfpidep1$(O) cpfpmdep1$(O) cpfcribds$(O)\
```

```
.....

HISTOR = cpglobals$(O) cprfunch$(O) cprdorech$(O) cprdesph$(O) cpfcdeph$(O)\

      cpfcrech$(O) cpfcdefh$(O) cpfinicia$(O) cpfpaifil$(O) cprmapimph$(O)\

.....

$Psysexten$(O) $Psysform$(O) $Psysmes$(O) \

 $Psystempo$(O) $Psysutil$(O) $Psysverlf$(O) $Psysverso$(O)

bb : ser$(EXEC) explor$(EXEC) principal$(EXEC) orcam$(EXEC) altrev$(EXEC) banco$

(EXEC) contabi$(EXEC) tesou$(EXEC) reqfund$(EXEC) histor$(EXEC) novano$(EXEC) ex

torno$(EXEC);

novano$(EXEC) : $(NOVANO)

          $(COMP) $(NOVANO) $(LIBS) $(LIBC) $(EXECFLAG) novano$(EXEC);

histor$(EXEC) : $(HISTOR)

          $(COMP) $(HISTOR) $(LIBS) $(LIBC) $(EXECFLAG) histor$(EXEC);

explor$(EXEC) : $(EXPLOR)

.....

principal$(EXEC) : $(PRINC)

 $(COMP) $(PRINC) $(LIBS) $(LIBC) $(EXECFLAG) principal$(EXEC);

orcam$(EXEC) : $(ORCAM)

 $(COMP) $(ORCAM) $(LIBS) $(LIBC) $(EXECFLAG) orcam$(EXEC);

ser$(EXEC) : $(SER)

 $(COMP) $(SER) $(LIBS) $(LIBC) $(EXECFLAG) ser$(EXEC);

msgmsg$(O) : msgmsg.4gl

          $C -$(FLAGS) msgmsg.4gl;

          rm *.c *.ec

cpflibcat$(O) : cpflibcat.4gl

          $C -$(FLAGS) cpflibcat.4gl;

          rm *.c *.ec

cpglobals$(O) : cpglobals.4gl

          $C -$(FLAGS) cpglobals.4gl;

          rm *.c *.ec
```

```
altrmain$(O) : altrmain.4gl
        $C -$(FLAGS) altrmain.4gl;
        rm *.c *.ec
princ$(O) : princ.4gl
        $C -$(FLAGS) princ.4gl;
        rm *.c *.ec
.....
cpftottes$(O) : cpftottes.4gl
 $C -$(FLAGS) cpftottes.4gl;
 rm *.c *.ec
cprgevnp$(O) : cprgevnp.4gl
 $C -$(FLAGS) cprgevnp.4gl;
 rm *.c *.ec
#### orcamentos##########
#cpglobals$(O) : cpglobals.4gl
orcamento$(O) : orcamento.4gl
        $C -$(FLAGS) orcamento.4gl;
        rm *.c *.ec
cpfcdef0$(O) : cpfcdef0.4gl
    $C -$(FLAGS) cpfcdef0.4gl;
    rm *.c *.ec
cpmorc$(O) : cpmorc.4gl
.....
cpfconchq$(O) : cpfconchq.4gl
 $C -$(FLAGS) cpfconchq.4gl;
 rm *.c *.ec
cprguidep$(O) : cprguidep.4gl
    $C -$(FLAGS) cprguidep.4gl;
    rm *.c *.ec
#####contab ######
```

```
#c4gl cpglobals.4gl

cpfpagddia$(O) : cpfpagddia.4gl

    $C -$(FLAGS) cpfpagddia.4gl;

    rm *.c *.ec

cpfcontinu$(O) : cpfcontinu.4gl

    $C -$(FLAGS) cpfcontinu.4gl;

    rm *.c *.ec

cprrdia$(O) : cprrdia.4gl

    $C -$(FLAGS) cprrdia.4gl;

    rm *.c *.ec

.....

relogio$(O) : relogio.4gl

    $C -$(FLAGS) relogio.4gl;

    rm *.c *.ec

sysdivide$(O) : sysdivide.4gl

    $C -$(FLAGS) sysdivide.4gl;

    rm *.c *.ec
```

Anexo V

1) GLOSSÁRIO

Aplicativo

É todo arquivo executável que possa rodar sob um sistema operacional. O aplicativo realiza uma tarefa por si só, ou seja, não depende de outros programas para funcionar. Por exemplo, o arquivo winword.exe é o Microsoft Word, aplicativo utilizado para edição de textos. Os arquivos de extensão *.doc, por sua vez, são os documentos criados pelo Word, e não realizam uma tarefa por si só. Sendo assim, arquivos *.doc não constituem um aplicativo.

ASP

Active Server Pages: são páginas criadas dinamicamente pelo servidor *Web*, orientado por um programa em *VBscript* ou *Javascript*. Quando um *browser* solicita uma página do tipo ASP, o servidor constrói uma página HTML e a envia ao *browser*. A diferença entre uma página ASP e um documento HTML clássico, é que o segundo corresponde a um documento estático, que já se encontra no servidor no formato que em que será exibido no navegador. A página ASP, ao contrário, não existe no servidor: é montada a partir de uma solicitação específica. Pode ser, por exemplo, um documento HTML criado como resultado de uma pesquisa num banco de dados.

Back-end

Um processador escravo que realiza alguma tarefa especializada, como a aceleração do acesso a um banco de dados.

Browser

A palavra "browse" significa examinar casualmente e um *browser* é um programa que permite a navegação da Internet e a visualização das páginas *Web*. Os *browsers* mais difundidos são, *Netscape Communicator* e o *Internet Explorer*.

CGI

Common Gateway *Interface*. É uma especificação que gerencia a transferência dados entre um servidor *Web* e o *browser* do usuário. Normalmente o programa CGI é executado no servidor, em resposta a uma solicitação do *browser* (por exemplo, quando se preenche um formulário). O código CGI pode ser escrito em diferentes linguagens, entre as quais C, *Perl*, *Java* ou *Visual Basic*. Outras formas de fazer esse processamento utilizam programas que rodam na máquina cliente: *AppletsJava*, Javascripts ou controles *ActiveX*.

Ciclo de CPU

É a menor unidade de tempo reconhecida pela CPU. Em geral, alguns centésimos-milionésimos de segundo. O termo também é usado com referência ao tempo necessário para que a CPU processe a instrução mais simples.

Cliente/servidor

Uma forma de estruturação das redes locais que utiliza o princípio da distribuição da inteligência, tratando tanto o servidor quanto as estações de trabalho como equipamentos inteligentes e programáveis, permitindo a exploração plena de toda a capacidade instalada da rede.

Consulta (*Query*)

Qualquer solicitação de pesquisa no banco de dados. Uma consulta típica pode ter a seguinte descrição: exibir todas as requsições que estão por cabimentar, no Sis-Pública.

CPD

Central de Processamento de Dados.

Extranet

Rede de negócio para negócio que une empresas parceiras por meio de suas intranets. Por usar os padrões abertos da Internet, os parceiros não precisam usar o mesmo sistema operacional, *hardware* ou *browser*.

Front End

Em geral, um computador ou unidade de processamento que gera e manipula dados antes que outro processador o receba.

GUI

Graphical User Interface. Usando imagens mais que apenas palavras para representar entradas e saídas de um *software*, a GUI é desenvolvida sob sistemas baseados em janelas, utilizando *ícones*, botões, caixas de diálogo, etc., que permitem ao usuário o controle das funções de um programa.

Hardware

Componentes físicos de um ambiente computacional (CPU, discos, *Modem*, cabos, etc.).

Hiperlink

Sequência de código que permite vincular um documento ao outro.

HTML

Hypertext Markup Language: formato para apresentação de informações em sistema de hipermídia, que inclui recursos de hipertexto (texto que usa *hiperlinks*), animação, som, formulários para captação de dados, etc., usado na *Web,* e interpretado por um *browser*.

Ícone

Pequenas imagens que sugerem a actividade a ser realizada.

Interface

Jargão de informática utilizado para designar a fronteira através da qual dois sistemas se comunicam. Pode ser um conector de *hardware* usado para conexão com outros equipamentos, ou uma convenção para a comunicação entre dois *softwares*. Normalmente é usado para designar todo o conjunto envolvido, *hardware* ou *software*, na comunicação entre um usuário e o computador.

ISDN

Sigla de *Integrated Services Digital Network*, ou rede digital de serviços integrados (RDSI). Trata-se de um padrão de comunicação para o envio de voz, dados e vídeo em linhas telefónicas digitais que utiliza a mesma estrutura de cabos da telefonia convencional. O ISDN exige cabeamento metálico e *Modems* especiais. Trabalha com dois canais de 64 Kpbs, o que permite transmitir a 128 Kpbs.

Java

Veja Linguagem de programação

Javascript

Veja Linguagem de programação

Jscript

Script Java. Veja Linguagem de programação.

Linguagem de programação

Linguagem formal na qual programas de computador (*softwares*) são escritos. A definição de uma linguagem particular consiste de sua sintaxe (como os vários símbolos podem ser combinados) e semântica (o significado das construções da linguagem).

Loop

Na programação, um conjunto de instruções executadas repetidamente por um número fixo de vezes ou até que alguma condição se torne verdadeira ou falsa.

Mainframe

São computadores com grande capacidade de processamento, para sua época, de alto custo, utilizado em empresas.

Modelo Relacional

Um modelo de dados no qual estes são organizados em relações (tabelas).

Monitor

É um periférico de saída e é o principal meio para visualização do usuário quando este utiliza um micro-computador. É neste dispositivo que normalmente é apresentada a *interface* para o usuário. Tradicionalmente é construído como um tubo de raios catódicos e componentes electrónicos associados, mas actualmente também são disponíveis em outras modalidades (consulte LCD, tela de gás-plasma, tela de matriz activa, tela de matriz passiva).

Multimídia

Recurso computacional que permite utilizar texto, imagem, som e animação para maior interacção com o usuário.

Palmtops

Computadores de dimensões semelhantes às de agendas electrónicas, incorporando recursos (processador de textos, planilhas electrónicas, banco de dados, acesso a *Web*, videoconferência, etc.) que eram habitualmente encontrados apenas em micro-computadores de maior porte.

PC

Acrónimo de *personal computer* (computador pessoal).

Periféricos

Na informática, um termo usado para indicar dispositivos como unidades de disco, impressoras, modems conectados a um computador e controlados por seu processador.

Perl

Veja Linguagem de programação

Plataforma

A tecnologia fundamental em que se baseia a construção de um sistema de computador. É a soma do hardware com o software básico, pois ambos fornecem suporte às aplicações.

RTF

Acrónimo de Rich Text Format, usado para transferência de documentos de texto formatados entre aplicações – inclusive entre aplicações executadas em plataformas diferentes.

SGML

Standard Generalized Markup Language. Linguagem genérica para formatação de documentos. A XML corresponde a uma versão reduzida do SGML aplicado somente à *Web*.

Sistema

Assume-se com sistema informatizado.

Sistema de informação

É um conjunto de pessoas, actividades, dados, redes, e tecnologia que são integrados com o propósito de auxiliar/melhorar as operações do dia a dia dum negócio.

Sistema informatizado

É a representação do sistema de informação usando as tecnologias computacionais

Sistema operacional

Software que controla o funcionamento de um equipamento computacional.

Software

Uma série de instruções para o computador que executam uma tarefa particular.

SQL

Structured Query Language, ou linguagem estruturada de consultas. Criada pela *IBM*, é uma ferramenta para extrair informações de bases de dados.

Tag

Códigos de formatação usado em documentos HTML para instruir o browser sobre a forma de apresentação de textos e gráficos numa *homepage*.

TCP/IP

Sigla de *Transmission Control Protocol/Internet Protocol*. Protocolo usado na comunicação entre computadores de redes diferentes. Compõe uma suíte de protocolos (HTTP, FTP, NNTP e mais de 100 outros) que formam a espinha dorsal dos padrões da *Internet*. O IP garante o endereçamento de todas as máquinas na *Internet* e o encaminhamento das mensagens entre essas várias máquinas.

URL

Uniform Resource Locator - endereço de uma página Internet (endereço de uma página *Web*, WWW). O endereço geralmente inicia com a sintaxe http://www.

WAP

A sigla WAP significa Wireless Application Protocol. É um conjunto de especificações que define um ambiente semelhante à *Web*, mas que funciona em redes de aparelhos sem fio e em velocidades mais baixas, como é o caso dos telefones celulares. Utilizando WAP você pode acessar sites na Internet que tenham sido construídos especificamente para a tecnologia *wireless* (utilizando o WML). Esses sites estão crescendo em número muito rapidamente.

Web

Sistema de informação distribuído, de arquitectura cliente-servidor, originado dos Laboratórios de Física de Alta Energia em Genebra, Suíça. No início dos anos 1990 a *Web* começou a espalhar-se pelo mundo e em 1993 passou a compartilhar informações utilizando a estrutura física da Internet.

WML

É uma linguagem de programação similar à HTML. A Wireless Markup Language (WML), baseada no padrão XML (Extensible Markup Language), é lida e interpretada por um microbrowser (micro-navegador) instalado num dispositivo WAP (um celular). Há, para esta linguagem, novos tipos de objectos. O WBMP (Wireless BitMap), por exemplo, é um gráfico em preto e branco que pode ser mostrado nos dispositivos WAP.

2) Abreviaturas e Termos Comuns

BDS	J's **B**usiness **D**evelopment **S**uite
BDL	Four J's **B**usiness **D**evelopment **L**anguage
DVM	Four J's **D**ynamic **V**irtual **M**achine
HFE	Four J's **H**TML **F**ront **E**nd

JFE	Four J's **J**ava **F**ront **E**nd
ODI	Four J's **O**pen **D**atabase **I**nterface
UC	Four J's **U**niversal **C**ompiler
WFE	Four J's **W**indows **F**ront **E**nd
4GL	**F**ourth **G**eneration **L**anguage
I4GL	**I**nformix **F**ourth **G**eneration **L**anguage
ALS	**A**sian **L**anguage **S**upport
GUI	**G**raphical **U**ser **I**nterface
UIBC	**U**ser **I**nterface **B**aseado em **C**aracteres
GLS	**G**lobal **L**anguage **S**upport
NLS	**N**ational **L**anguage **S**upport
OS	**O**perating **S**ystem
RDBMS	**R**emote **D**ata**B**ase **M**anagement **S**ystem
IDE	**I**ntegrated **D**evelopment **E**nvironment
Sis-Pública	**Sis**tema de Informação para a Gestão de Finanças **Públicas**