Sistemas de Informação para a Web: Arquitecturas Aplicacionais

Alberto Silva
IST/INESC
Rua Alves Redol, 9, 1000 Lisboa
Alberto.Silva@acm.org

Miguel Mira da Silva
IST/INESC
Rua Alves Redol, 9, 1000 Lisboa
mira-da-silva@ip.pt

José Delgado
IST/INESC
Rua Alves Redol, 9, 1000 Lisboa
Jose.Delgado@inesc.pt

1. SUMÁRIO

Propõe-se e expõe-se nesta comunicação um modelo de classificação e identificação das principais arquitecturas de sistemas de informação para a Web, designadamente baseadas em actividades computacionais (1) centradas no servidor, (2) centradas no cliente, e (3) híbridas, suportadas por infraestruturas distribuídas.

1.1 Palavras-chave

Web, Arquitecturas de Sistemas de Informação, SIW, CGI, SSI, Java, ORB.

2. INTRODUCÃO

A Web é um dos principais serviços que mais contribui para o sucesso da Internet. Sendo originalmente um sistema hipermédia distribuído para navegação e consulta de informação fracamente estruturada, cedo começaram as primeiras pressões dos utilizadores e das organizações para introdução de novas capacidades e funcionalidades que já existiam nos sistemas de informação tradicionais, tais como formulários (para introdução de informação), e interligação a sistemas de bases de dados.

Com estas pressões, a Web evoluiu para um sistema "chapéu" de

variados sistemas de informação, mais ou menos especializados, mas acessíveis à escala mundial ou à escala interna (de um grupo restrito de utilizadores).

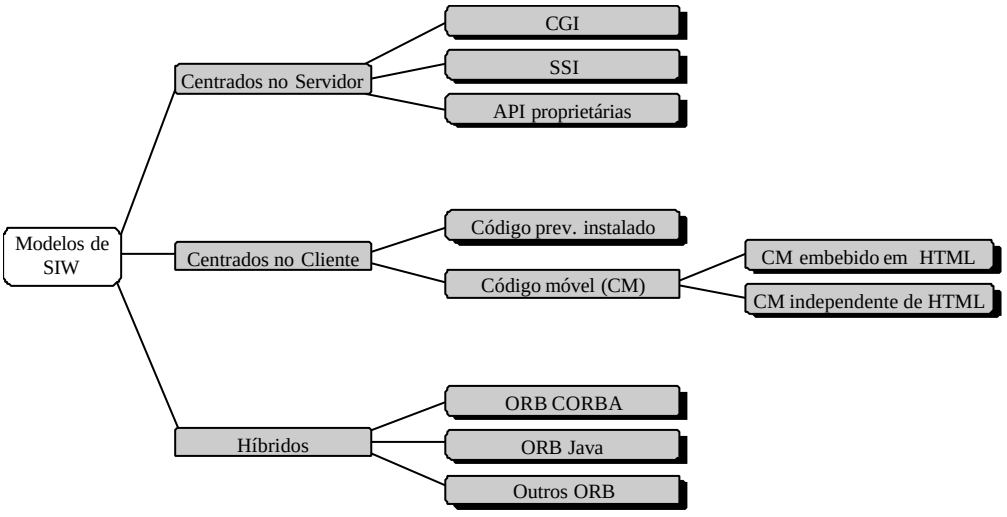
Figura 1: Tipos de Arquitecturas de SI para a Web

Actualmente a tecnologia para desenvolvimento e suporte de sistemas de informação para a Web (SIW) é foco de um enorme interesse e investimento porque permite conjugar as vantagens dos sistemas de informação tradicionais com as vantagens da Web. Por conseguinte, não é objectivo desta comunicação a descrição de ferramentas, ambientes de desenvolvimento e produtos particulares inerentes ao desenvolvimento de SIW, já que estes rapidamente são ultrapassados por novas gerações de produtos homólogos (serão apenas referidos, a título de exemplo, um ou outro sistema representativo de determinada aproximação).

Em vez disso, apresentam-se nesta comunicação em termos gerais os principais modelos, ou aproximações, inerentes às arquitecturas de sistemas de informação para a Web (vid. Figura 1).

A Secção 3 apresenta os modelos de SIW centrados no servidor, ou seja, sistemas de informação cuja actividade é realizada predominantemente por um ou vários processos executados na máquina servidor. A Secção 4 apresenta os modelos de SIW centrados no cliente, ou seja, sistemas de informação cuja actividade é realizada predominantemente na máquina cliente. A Secção 5 apresenta os SIW cuja actividade se encontra dividida entre as máquinas cliente e servidor e é suportada por uma infraestrutura distribuída.

Por fim, na Secção 6 apresentam-se os principais contributos e conclusões desta comunicação.



3. SIW CENTRADOS NO SERVIDOR

Todas os sistemas de informação com actividade centrada no servidor apresentam os seguintes pontos em comum:

- O facto do servidor Web não manter o estado (*stateless*) e do protocolo HTTP não manter a conexão (*connectionless*) exige que sejam as aplicações as responsáveis por tal facto. (Existem algumas técnicas de se simular a manutenção de estado sobre conexões HTTP. Desde informação que é passada nos próprios URL (e.g. nas componentes: *Query String* ou *Path Info*); passando por campos de elementos FORM, visíveis ou invisíveis.) Esta limitação também implica que os processos sejam de "vida-curta", ou seja, que tenham um ciclo de vida bem determinado, apenas circunscrito ao período temporal de um acesso desencadeado pelo utilizador.
- Os sistemas são basicamente geradores de documentos HTML (eventualmente com outras funcionalidade adicionais) que são criados dinamicamente à medida que se interactua com o utilizador [15].

Na sequência da realização dos primeiros SIW baseados no interface CGI (*Common Gateway Interface*), surgiram algumas variantes ao mecanismo básico, quer por motivos de simplicidade (mecanismo *Server Side Includes*), quer por razões de desempenho (mecanismo de API proprietárias providenciado pelos servidores Web), como será analisado em baixo.

3.1 Baseados no CGI

Os primeiros sistemas de informação para a Web (e.g., sistemas em que o autor esteve envolvido [15, 17, 14]) surgiram com a especificação CGI (*Common Gateway Interface*) [18, 13] que foi originalmente suportada no servidor Web do NCSA. O CGI surgiu como resposta à necessidade de estender a funcionalidade dos servidores Web de forma flexível e extensível. O objectivo era que o servidor providenciasse um interface simples que quaisquer outros processos pudessem interactuar de forma independente. Deste modo conseguiu-se responder a pressões de introdução no servidor Web de novas funcionalidades (e.g., acesso a sistemas de bases de dados específicas, conversores de dados) de uma forma extensível: apenas providenciando um interface ao nível de comunicação entre processos.

Os primeiros SIW baseados no CGI, consistiram essencialmente na conversão de diferentes protocolos e formatos de informação (e.g., X.500, WAIS) para formato HTML. Por esta razão foi habitual designá-los por "*gateways*" (conversores). Também foram designados de "*scripts*" pelo facto de terem sido (e continuarem a ser) desenvolvidos principalmente em linguagens de programação interpretadas (e.g., Perl, C-Shell) ao nível do sistema operativo.

A Figura 2 esquematiza a arquitectura de um SIW baseado no mecanismo CGI. Apresenta-se com maior ênfase (a cinzento) os blocos da aplicação propriamente dita. Por um lado o "`Processo CGI`", por outro um bloco não identificado ("`??`") propositadamente. Este bloco poderá corresponder a um processo mais específico (e.g., leitura de imagens de uma câmara de vídeo) ou mais geral (e.g., servidor de ficheiros, servidor de gestão de bases de dados, ou servidor de informação geográfica) com o qual o processo CGI poderá comunicar. Os restantes blocos – cliente e servidor Web – são gerais, são utilizados como componentes de suporte, e consequentemente não lhes será dada relevância nesta comunicação.

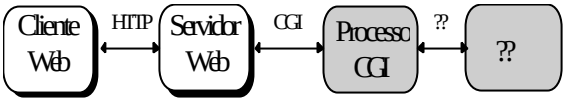


Figura 2: Arquitectura de SIW baseado no CGI

O processo CGI evolui, em cada acesso desencadeado remotamente pelo utilizador, ao longo de cinco fases

1. É criado pelo servidor Web.
2. Os dados (provenientes do servidor Web) são recebidos através do "canal" CGI (e.g., em Unix, através de variáveis de ambiente e do *standard input*).
3. Analisa os dados recebidos e verifica a sua validade.
4. Executa determinado processamento dependendo da informação recebida (e.g., pode estabelecer uma conexão a determinado SGBD e executar uma interrogação SQL).
5. Produz resultados (geralmente no formato HTML) para o "canal" CGI (e.g., em Unix, é o *standard output*), que serão lidos pelo servidor Web e redireccionados para o cliente.

3.2 Baseados no SSI

O mecanismo SSI (*Server Side Includes*) foi inicialmente proposto e implementado no servidor Web do NCSA [8] sendo actualmente um dos mecanismos mais populares e adoptados pela generalidade dos servidores Web.

A Figura 3 esquematiza o funcionamento genérico do mecanismo SSI que estende o HTML com elementos que são interpretados (dinamicamente) pelo próprio servidor Web.

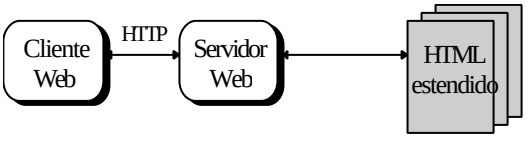


Figura 3: Arquitectura de SIW baseado no SSI

Primeiro é realizado o pedido de determinado documento HTML. O Servidor Web lê o ficheiro HTML

correspondente, faz a interpretação dos elementos estendidos, e substitui-os por informação correspondente. Como exemplos, tem-se: a data da última alteração do ficheiro; ou o resultado de uma interrogação SQL realizada. Por fim é enviado, para o cliente, o documento solicitado com as adições (*includes*) introduzidas. A figura dá ênfase (a cinzento) ao documento "`HTML estendido`" (i.e., documento HTML com elementos proprietários) pelo facto de ser apenas necessário construir-se essa teia de documentos de forma a definir-se o SIW.

Na Figura 4 ilustra-se o processo de interpretação e conversão dos elementos HTML estendidos realizado pelo servidor Web. O elemento `<!--#echo var= "LAST_MODIFIED"-->` é convertido para o texto correspondente à data corrente do sistema (por exemplo, "`1997/10/26`").

Compete ao servidor fazer a leitura do documento solicitado pelo cliente (`x.shtml`), detectar (pela extensão do ficheiro) que corresponde a um documento com elementos não standard (extensões) e proceder às suas substituições (`x.html`). Todas as extensões para o servidor são formatadas como comentários SGML (elemento `<!--" "-->`) dentro do documento original (`x.shtml`) para o caso em que o documento possa ser consultado pelo cliente sem passar pelo mecanismo de substituição realizado pelo servidor.

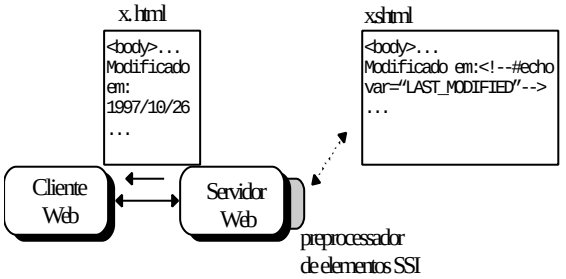


Figura 4: Processo de interpretação e conversão no mecanismo SSI

Este mecanismo original do servidor Web do NCSA desencadeou o aparecimento de mecanismos equivalentes, mas mais potentes e evoluídos. O seu objectivo genericamente é permitir que qualquer utilizador, sem ter que utilizar qualquer linguagem de programação – como acontece na abordagem CGI –, possa desenvolver SIW simples e de forma relativamente rápida.

O htmlscript da Volant [19], por exemplo é um sistema (é uma aplicação CGI) que funciona como preprocessador convertendo ficheiros em formato htmlscript em ficheiros com elementos HTML standard. O htmlscript é uma extensão da linguagem HTML que introduz novos elementos (*tags*), tais como `<if>`, `<evaluate>`, `<export>`, `<import>`, `<let>`, `<hide>`, `<exit>` e `<debug>`.

O Internet Database Connector [5], da Microsoft, é outro exemplo da adopção do objectivo original do SSI. O Internet Database Connector é uma aplicação (`httpodbc.dll`) que usa uma especificação estendida do HTML (os ficheiros com extensão `HTX`) que conjuntamente com os ficheiros `IDC` (onde é mantida a informação sobre o estabelecimento de conexões a fontes de dados ODBC e como executar determinada instrução SQL sobre a referida fonte de dados) permite o desenvolvimento de aplicações simples. Esta tecnologia foi posteriormente substituída pela homóloga Active Server Pages [20], que consiste na interpretação de ficheiros `ASP` (que incluí elementos HTML com código em BasicScript) pelo servidor Web da Microsoft.

3.3 Baseados em Servidores Especializados

De modo a providenciar o desenvolvimento de SIW com um nível de desempenho superior ao patente no CGI e com funcionalidades superiores ao SSI, surgiram servidores Web, que incluíam adicionalmente funcionalidades especializadas. A Figura 5 ilustra a componente do servidor Web que inclui, no mesmo programa, uma determinada funcionalidade particular.

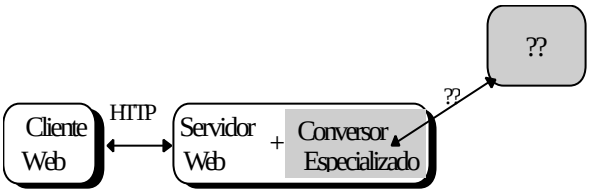


Figura 5: Arquitectura de SIW baseado em servidores especializados

Esta aproximação, apresenta um bom nível de desempenho e um elevado nível de integração com serviços especializados, tipicamente de acesso a bases de dados. Todavia, baseia-se numa solução proprietária e por isso fechada e pouco flexível. Rapidamente esta aproximação foi suplantada ou complementada por servidores Web, com interfaces proprietárias conforme apresentado na secção seguinte.

3.4 Baseados em Interfaces Proprietárias

Levados pelas questões do baixo nível de desempenho das aplicações baseados no interface CGI, e pela vontade de desenhar servidores mais extensíveis e configuráveis, mas ainda assim fáceis de utilizar, os principais fabricantes da indústria introduzem nos seus servidores uma interface ao nível de programação (API) [21, 22].

As aplicações baseadas nas API dos servidores Web são carregadas conjuntamente no momento do "arranque" (*boot*) do servidor ou em tempo de execução à medida que vão sendo invocadas. Todavia ficam residentes em memória no mesmo espaço de endereçamento do processo servidor. Não se evita o "ciclo de vida curto" das aplicações baseadas no CGI, mas evita-se a fase de criação de cada processo (com o respectivo carregamento em memória, mudança de registos, alocação de recursos,

etc.). O requisito para que os servidores Web providenciem API depende única e exclusivamente do suporte de programas dinamicamente carregáveis e partilháveis que o sistema operativo providencie. Por exemplo, em ambiente Windows existe o formato de ficheiro DLL (*Dynamic Link Library*).

A Figura 6 esquematiza o modelo abstracto das aplicações baseadas neste modelo.

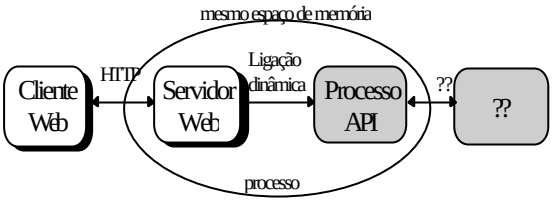


Figura 6: Arquitectura de SIW baseado em API proprietárias dos servidores

As aplicações baseadas neste modelo não evitam o padrão computacional "pedido-resposta" do modelo CGI, tal como todo o mecanismo de descodificação de informação passada nos URL; nos campos dos elementos FORM; e a resposta em formato HTML.

Mas agora em cada acesso o sistema evolui nas seguintes fases:

1. O servidor detecta (com base na informação passada no URL) que deverá invocar determinada função de uma das suas aplicações externas (que foram ou serão previamente carregadas).
2. O servidor invoca a referida função passando-lhe todos os parâmetros necessários. Em geral esta função é genérica e única, tendo um parâmetro que consiste numa estrutura com a maioria da informação necessária e adicionalmente num conjunto de funções invertidas (*callbacks*) com as quais a aplicação poderá invocar o servidor de forma a solicitar informação extra e/ou a providenciar outras funcionalidades.
3. A referida função deverá validar os parâmetros recebidos e executar o devido processamento. Caso necessite de aceder a informação que não lhe tenha sido passada nos seus argumentos, utilizará as funções invertidas providenciadas de modo a solicitá-las ao servidor.
4. A referida função termina a sua execução retornando o seu resultado (geralmente no formato HTML), que será lido pelo servidor e redireccionados para o cliente. O processo de escrita da informação também utiliza uma das funções invertidas providenciadas pelo servidor.

Actualmente cada fabricante providencia uma API específica do seu servidor. Contudo o NSAPI (*Netscape API*) [21] da Netscape e o ISAPI (*Internet Server API*) [22] da Microsoft são os mais populares.

4. SIW CENTRADOS NO CLIENTE

À medida que se construíam os primeiros SIW, baseados essencialmente no mecanismo CGI, começou-se a perceber as reais limitações da tecnologia Web relativamente à construção de SI:

- Baixo nível de interacção homem-máquina
- Baixo nível de desempenho
- Dificuldade em suportar transações complexas

Os utilizadores habituados ao nível de interacção homem-máquina patenteado na generalidade dos SI das suas organizações, baseados no modelo cliente/servidor e em interfaces gráficas multi-modais, exigiram mais e melhores capacidades para os seus SIW.

Podem resumir-se os modelos de SIW centrados no cliente a dois grupos principais. Por um lado, o modelo de aplicações baseado em código previamente carregado. Por outro, o modelo de aplicações baseado em código móvel. (A distinção entre um grupo e o outro nem sempre é clara como veremos.)

4.1 Baseados em Código Previamente Instalado

No seguimento do mecanismo de **invocação externa de programas** suportada pelas primeiras versões do cliente Web da Netscape (em que programas externos são lançados/invocados pelo cliente Web, tendo em conta o tipo MIME do documento recebido e uma tabela de configuração apropriada), surgiu o conceito de **invocação interna de programas**. Este mecanismo consiste na invocação de programas executados no contexto computacional do cliente, no mesmo espaço de endereçamento, e integrado no mesmo sistema de janelas gráficas.

Na aproximação da invocação externa de programas, o cliente detecta, pela indicação do formato MIME do documento recebido (e.g., `text/doc`), qual a aplicação a invocar (e.g., MS-Word). Invoca a aplicação específica passando-lhe o documento previamente recebido e perde a ligação e o controlo com esse documento e correspondente aplicação.

De modo a permitir um melhor nível de coesão e interacção entre o cliente Web e as aplicações específicas, e de não se perder o contexto da Web, foram concebidos originalmente os protocolos CCI (*Common Client Interface*) [9] e CCI++ (*Constituent Component Interface++*) [1]. A Netscape, baseando-se nos objectivos e capacidades patenteadas na tecnologia CCI++, incorpora nos seus clientes um API que permita a integração e execução de aplicações externas, as quais são designadas por "*plug-ins*" [10].

Os *plug-ins* consistem em aplicações (módulos de ligação dinâmica) previamente instaladas e responsáveis pelo tratamento de um ou mais tipos MIME. São invocadas e executadas dinamicamente pelo cliente Web quando este recebe um documento do tipo MIME associado.

A Figura 7 realça (a cinzento) as seguintes componentes particulares: (1) a aplicação previamente instalada no contexto do cliente Web; (2) os documentos MIME; e (3) a aplicação executada no contexto do servidor Web e que é responsável pela produção e envio dos documentos MIME envolvidos

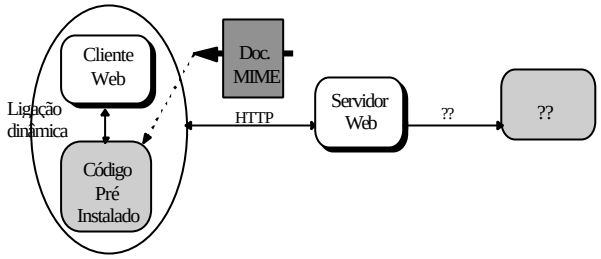


Figura 7: Arquitectura de SIW baseado em código previamente instalado

Em geral as aplicações que adoptam este modelo são relativamente genéricas e produzidas por *software houses* especializadas. A generalidade dos exemplos consistem em visualizadores e editores da maioria dos formatos de informação multimédia, tais como: Acrobat, VRML, Director, MPEG2, e PowerPoint. Outros exemplos são editores de folhas de cálculo e máquinas virtuais de linguagens/formatos particulares, tais como a máquina virtual de Java ou a máquina virtual do sistema de código móvel Omniware da Colusa Software [4].

4.2 Baseados em Código Móvel

A abordagem apresentada anteriormente permite um razoável grau de coesão e de integração de aplicações externas com clientes Web. As aplicações encontram-se associadas a tipos de documentos específicos, relativamente aos quais são excelentes visualizadores e eventualmente editores com suporte hipermédia. Contudo esta aproximação apresenta limitações ao nível da portabilidade (o mesmo *plug-in*, para diferentes versões de sistemas operativos, teria de ser consideravelmente redefinido já que a própria API providenciada pela Netscape não é idêntica para os diferentes sistemas operativos) e da versatilidade das aplicações finais. Adicionalmente, exige uma prévia instalação das referidas aplicações e um consequente esforço humano de actualização de versões.

Uma abordagem mais abrangente e geral de desenvolvimento de SIW é baseada em sistemas de código móvel [23, 2, 24].

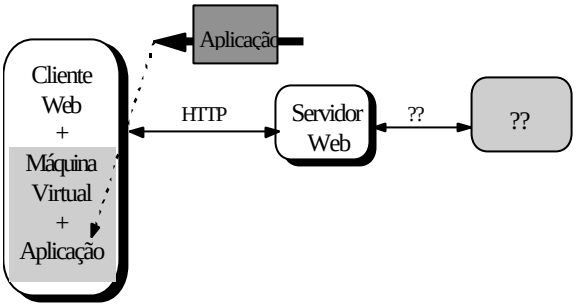


Figura 8: Arquitectura de SIW baseado em código móvel

Conforme se observa na Figura 8, o código específico da aplicação encontra-se mantido e gerido no servidor Web (componente "`??`"), mas pode ser transportado (transferido), a pedido, para a máquina cliente de modo a ser executado nesse contexto (componente "`máquina virtual`"). Nalguns casos, o cliente providencia uma máquina virtual responsável pela interpretação/execução segura do código recebido. Noutras situações, o código é executado directamente pela máquina real.

Existem basicamente duas aproximações de SIW baseados em código móvel: (1) código móvel embebido no documento HTML; e (2) código móvel independente/separado do documento HTML.

4.2.1 Código Móvel Embebido no Documento HTML

Na aproximação de código móvel embebido no documento HTML, o cliente Web, para além da capacidade de interpretação e apresentação dos elementos HTML, faz também a interpretação e execução do código embebido, que se encontra em código fonte. Esta é a aproximação adoptada pela generalidade das linguagens de *scripting:* JavaScript [11], VBScript [7], Tcl [12], Obliq [3], etc.

Este tipo de aproximação tem, entre outros, dois objectivos principais: permitir a construção de SIW relativamente simples e pequenos; e integrar convenientemente documentos HTML com outras tecnologias (tais como applets Java, controlos Active-X, e *plugs-ins*) funcionando como tecnologia orquestradora desses diferentes componentes.

Actualmente as linguagens que parecem vir a dominar nesta área são o JavaScript e o VBScript. O JavaScript foi desenvolvido conjuntamente pela Netscape e Sun, e apesar do nome sugerir uma semelhança entre Java e o JavaScript, é totalmente distinto da linguagem Java. Por seu lado a Microsoft, lançou o Visual Basic Scripting Edition, (designado abreviadamente por VBScript), que corresponde a um subconjunto seguro da linguagem de configuração das aplicações de escritório (*Office*) conhecida por Visual Basic for Applications (que por sua vez é um subconjunto do Visual Basic).

Ambas apresentam um conjunto de características comuns: consistem em linguagens relativamente reduzidas, seguras e independentes da plataforma. Ambas

providenciam um hierarquia estática de objectos (*built-in object hierarchy*) de interacção homem-máquina (e.g., manipulação das janelas do cliente, formulários e seus componentes, e suporte para tratamento das âncoras de hipertexto) com um conjunto conhecido de propriedades (e.g., cor, dimensão, texto de ajuda, etc.) e de métodos, alguns dos quais correspondentes a rotinas de eventos predeterminados (e.g., `OnClick`, `OnFocus`, `OnChange`, `OnSubmit`). Também oferecem suporte para configuração (*scripting*) de *plug-ins*, applets Java e controlos Active-X.

4.2.2 Código Móvel Independente do Documento HTML

Na **aproximação de código móvel independente do documento HTML**, a máquina virtual faz a interpretação do código referido num determinado documento HTML, mas que se encontra num ficheiro separado (gerido e mantido na máquina servidor) e independente do documento HTML original. Como tal, o código pode ser referenciado mais que uma vez dentro de um documento e em mais que um documento HTML. O código é transferido e interpretado em formato de texto, *byte-code* ou código máquina (vid. Figura 9). Esta é a aproximação adoptada, por exemplo, pelo Java e pelos controlos activos (Active-X) da Microsoft. Também já foram realizadas experiências, segundo este modelo, com algumas das linguagens referidas na aproximação anterior, tais como Tcl ou Obliq.

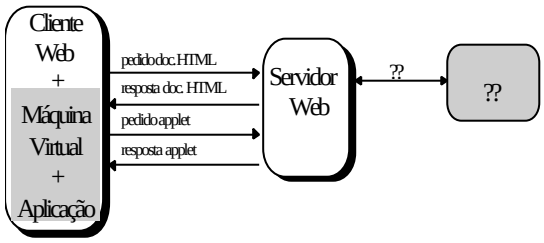


Figura 9: Código móvel, independente do documento HTML

No caso do Java, o código móvel consiste numa mini-aplicação (applet) definida em código intermédio (*Java byte code*), o qual é executado pela máquina virtual Java que existe no contexto de cada cliente Web.

A Microsoft lançou a sua tecnologia concorrente, designada por Active-X [6], que assenta na tecnologia já existente de controlos OCX/OLE com suporte adicional para acesso à Internet. Os controlos Active-X podem ser referidos num documento HTML, sendo de seguida pesquisados no registo Windows da máquina cliente, ou, caso não existam, carregados a partir de um servidor Web remoto e instalados automaticamente antes de serem executados. Os dados de inicialização do controlo podem encontrar-se dentro do documento HTML, ou serem carregados a partir do servidor Web remoto. Os controlos activos correspondem basicamente à tecnologia dos controlos OLE e como tal pressupõem as seguintes características (que são simultaneamente vantagens e desvantagens, dependendo do ponto de vista da análise): podem ser utilizados tanto no contexto de um cliente Web, como em outras aplicações Windows existentes; são desenvolvidos em qualquer linguagem que suporte a construção de controlos OLE (e.g., VC++, Delphi); e são executados directamente em código nativo, o que por um lado apresenta níveis de desempenho optimizados, mas por outro levanta o problema da portabilidade e da gestão e manutenção de inúmeras versões do mesmo código para plataformas distintas. A questão da segurança é tratada ao nível da certificação dos componentes: uma componente/mini-aplicação é segura se for certificada por uma entidade independente em que o utilizador confie.

Por estas razões o paradigma dos controlos activos apresenta uma situação intermédia entre a aproximação de código previamente instalado e a aproximação de código móvel independente do documento HTML. A razão fundamental de o integrar neste último modelo deveu-se ao facto dos controlos activos não serem activados pelo cliente, consoante o tipo MIME da informação recebida, mas sim, quando são referidos explicitamente dentro de um documento HTML.

5. SIW SUPORTADOS POR INFRAESTRUTURAS DISTRIBUÍDAS

O modelo de aplicações cuja actividade é realizada predominantemente no cliente, veio solucionar ou pelo menos atenuar os problemas de desempenho e de interacção homem-máquina patenteados no modelo de aplicações centrado no servidor. Todavia, manteve-se o problema do ineficiente suporte do protocolo HTTP a transacções complexas. O HTTP baseia-se no padrão pedido-resposta e consequentemente não se adapta a sistemas e serviços específicos, tais como vídeo ou audio em tempo real, ou sistemas de bases de dados que exigem transacções complexas, longas e eventualmente distribuídas.

Desta forma, começaram a surgir propostas que aqui serão designadas por soluções híbridas. Nesta aproximação, conforme ilustrado na Figura 10, o cliente e o servidor Web são utilizados no início da interacção, apenas para estabelecimento de ligação e eventual transferência de parte da aplicação envolvida. (No caso de *plug-ins* poderá nem ser necessário uma ligação inicial ao servidor Web. E no caso de código móvel apenas para carregamento do applet correspondente). Seguidamente, a aplicação (applet, *plug-ins*, ou controlos Active-X) estabelece uma conexão a um servidor proprietário especializado. Consequentemente, a aplicação perde o contacto com a Web, e do cliente Web apenas é utilizado a sua máquina virtual ou ambiente de execução.

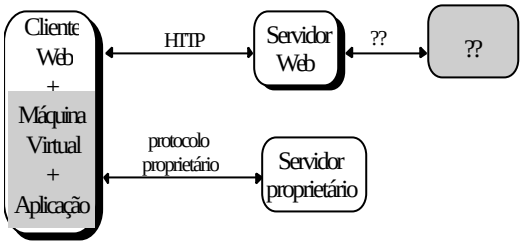


Figura 10: Arquitectura de SIW baseado em aproximações híbridas

Afim de se evitar as questões de integração e de interoperabilidade patentes na aproximação híbrida, em que clientes e servidores especializados acordam, de forma bilateral, um protocolo proprietário de comunicação é proposta uma abordagem suportada por uma infraestrutura de comunicação geral e comum (vid. Figura 11).

A ideia fundamental, é que a tecnologia Web seja utilizada, por um lado, como mecanismo de acesso uniforme a recursos e serviços; e, por outro lado, como mecanismo de interacção homem-máquina consistente baseado, por exemplo, em documentos HTML ou em applets Java suportados por bibliotecas standard (e.g., AWT, Swing).

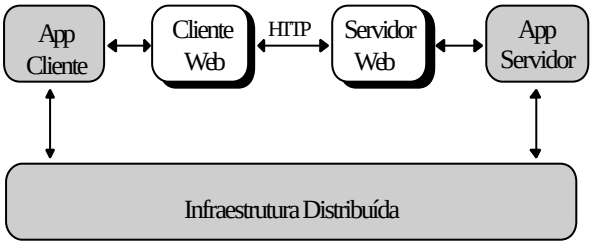


Figura 11: Arquitectura de SIW suportado por uma infraestrutura distribuída comum

Apresenta-se de seguida duas aproximações principais de aplicações suportadas por infraestruturas de comunicação.

5.1 Infraestrutura Baseada em CORBA

Uma aproximação de aplicações distribuídas integradas na Web suportada pela arquitectura CORBA é ilustrada na Figura 12.

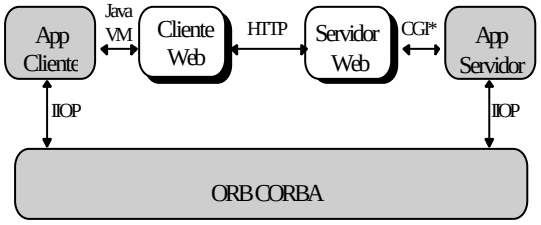


Figura 12: Arquitectura de SIW suportado por uma infraestrutura baseada em CORBA

O CORBA, (*Common Object Request Broker Architecture*) é um standard que permite a comunicação e interoperação entre objectos distribuídos de forma independente da linguagem de programação, do sistema operativo, e da arquitectura de suporte. Existem três componentes fundamentais que constituem, da perspectiva do programador, o núcleo do CORBA [25, 16]:

- IDL (*Interface Definition Language*): É a linguagem usada para se especificar a interface de um objecto CORBA.
- IIOP (*Internet Inter-Orb Protocol*): É um protocolo que especifica o formato de rede, independente da linguagem, usado para codificação e envio de mensagens entre objectos CORBA.
- IOR (*Interoperable Object Reference*): É o formato, independente da linguagem, usado para codificar uma referência de um objecto CORBA (que inclui o endereço IP, o número do porto, e o nome da classe do objecto respectivo).

As aplicações baseadas no CORBA envolvem geralmente as seguintes tarefas/fases (vid. Figura 13):

1. A interface do objecto CORBA que se pretende tornar acessível é escrita na linguagem IDL.
2. A partir do ficheiro IDL, anteriormente escrito, são gerados adaptadores (*stubs*) de cliente e de servidor para uma determinada linguagem de programação.
3. A classe do objecto é construída a partir do adaptador de servidor gerado anteriormente (basicamente pela introdução de código nos métodos gerados e pela definição de novos métodos, que obviamente não farão parte da interface pública).
4. Um objecto da classe anteriormente definida é criado no contexto de um servidor CORBA e é tornado acessível através de um serviço de nomes.
5. Um programa cliente obtém uma referência para o objecto CORBA através de um serviço de nomes (o programa cliente tem de possuir um adaptador de cliente da classe relativa ao objecto que pretende referenciar/invocar).
6. As mensagens são enviadas para o objecto CORBA através do adaptador de cliente que é responsável pela sua transmissão para o objecto remoto através do protocolo IIOP.

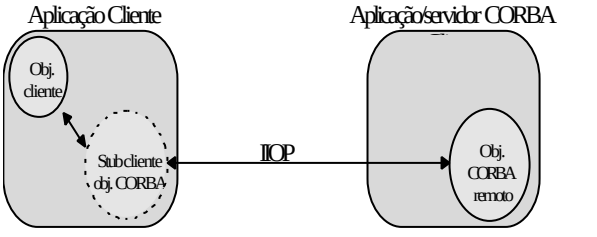


Figura 13: Invocação de método sobre um objecto CORBA remoto, via IIOP

Existem actualmente produtos que permitem que a partir de um cliente Web se aceda a objectos CORBA remotos através do protocolo IIOP. Existem nomeadamente duas aproximações

- O próprio cliente Web inclui (o código de) um ORB CORBA básico – esta solução é promovida designadamente pela Netscape que integra o VisiBroker nos seus próprios produtos.
- O ORB CORBA é carregado remotamente via HTTP, na forma de uma biblioteca de classes Java, no momento (ou a pedido) do carregamento de um applet Java. (Por motivos de desempenho e de forma a evitar-se o carregamento sistemático do ORB adoptam-se mecanismos de cache apropriados.) Esta solução é suportada pela generalidade dos fabricantes de ORB destacando-se entre outros: o OrbixWeb da Iona [26], o Voyager da ObjectSpace [28], ou o VisiBroker da Inprise [29].

Do lado da máquina servidor, a aplicação servidor específico pode aceder a objectos CORBA quer com base nos ORB anteriormente referidos, caso a aplicação seja em Java, quer quaisquer outros ORB que suportem qualquer outra linguagem de programação (e.g., C++, Cobol, Smalltalk).

5.2 Infraestrutura Baseada no Java

A grande virtude da existência de ORB CORBA para clientes Web (directamente, ou indirectamente através de bibliotecas de classes Java, como visto) é permitir a integração de aplicações e serviços legados na Web. No entanto, existem desvantagens na adopção de CORBA como ORB principal no desenvolvimento de aplicações baseadas em Java, entre outras [28]:

- As classes Java com suporte CORBA exigem modificações significativas às classes originais.
- O número de classes auxiliares de suporte (definição de adaptadores e de interfaces) é significativo – tipicamente quatro a cinco classes por classe original – o que exige um esforço adicional de gestão e manutenção das classes.
- O IDL não suporta todas as características da linguagem Java, designadamente: passagem de parâmetros por valor; múltiplos métodos com nomes idênticos; e herança de classes de excepção.
- O CORBA não define a semântica para a construção de objectos remotos, nem suporta carregamento dinâmico de classes, ou reciclagem de memória distribuída.

Ou seja, o CORBA devido aos seus objectivos de suporte de múltiplas linguagens torna-se uma tecnologia difícil de utilizar, nomeadamente em confronto com ORB concebidos especificamente para o desenvolvimento em Java. Surgiram, no contexto da interoperação de objectos Java, propostas de ORB caracterizados principalmente pela simplicidade e facilidade de programação, de entre os quais destacam-se o RMI da Sun [31] e o Voyager da ObjectSpace [27].

A Figura 14 ilustra o modelo geral desta classe de aplicações. Por confronto com a Figura 12, observa-se a mudança de ORB (passa de ORB baseado em CORBA para ORB baseado em Java), e consequente mudança de protocolo de comunicação entre objectos.

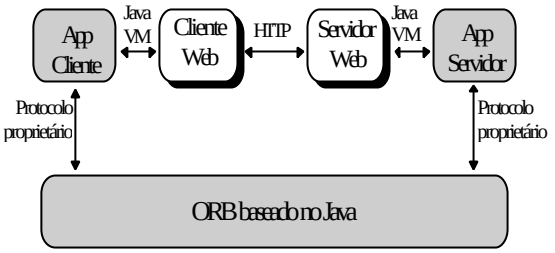


Figura 14: Arquitectura de SIW suportado por uma infraestrutura baseada em Java

Do lado da máquina cliente, a aplicação consiste num applet Java que é executado no contexto da máquina virtual do cliente Web. Para além da interacção homem máquina específica, o applet poderá solicitar funcionalidades providenciadas por outras aplicações (objectos) Java distribuídas no espaço gerido pela infraestrutura comum.

Do lado do servidor, o servidor Web pode ser ele próprio uma aplicação Java [30, 32], facilitando o mecanismo de comunicação entre processos (servidor Web e aplicações Java específicas, por exemplo servlets [33]). A aplicação específica pode exportar objectos Java que sejam acessíveis remotamente e/ou aceder a objectos Java exportados por outras aplicações. Todo este mecanismo é suportado de forma integrada com o ORB utilizado.

6. CONCLUSÕES

O principal contributo desta comunicação foi a sistematização e classificação dos principais modelos, ou aproximações, das arquitecturas de sistemas de informação para a Web.

Em termos gerais, as tecnologias que se encontram patentes no desenvolvimento dos sistemas de informação das organizações são principalmente: sistemas baseados em servidores de ficheiros partilhados, sistemas de objectos distribuídos, sistemas suportados por gestores de bases de dados, ou sistemas de processamento transaccional. Enquanto que as tecnologias associadas aos sistemas de ficheiros partilhados, de bases de dados e de processamento transaccional encontram-se maduras, testadas e adequadas à construção de sistemas de informação, respectivamente, das pequenas, médias e grandes organizações, já o mesmo não acontece com a tecnologia associada aos sistemas de objectos distribuídos – esta é uma das expectativas que a industria informática criou na primeira metade da década de 90 e que só agora começa a ser testada efectivamente.

Paralelamente surgiu o serviço WWW, originalmente entendido como um sistema hipermédia distribuído, de fácil navegação e procura de informação pouco estruturada. Este evoluiu rapidamente para um sistema "chapéu" de inúmeros e variados SI, e passou-se a designá-lo indiscriminadamente por "Web". Os modelos referidos no parágrafo anterior são primordialmente centrados nas organizações e nos seus recursos computacionais. Por outro lado, nos sistemas de informação baseados na Web o foco é centrado em grandes comunidades de utilizadores (eventualmente anónimos) e na computação baseada em redes de computadores, quer locais, quer em grande escala.

Assiste-se actualmente a um esforço de integração entre estes dois mundos, cujo objectivo último é o desenvolvimento dos sistemas de informação para a Web/Internet, bem como, para os contextos internos de organizações ou grupos restritos de organizações (i.e., para Intranets). Destacam-se especificamente os modelos centrados no servidor, centrados no cliente, e híbridos, suportados por infraestruturas de objectos distribuídos.

A aproximação centrada no servidor teve como principal contributo o facto de ter permitido a criação dos primeiros SIW de forma simples e efectiva. Contudo esta aproximação apresenta algumas limitações importantes: dificuldade de suportar transações complexas, baixo nível de desempenho, e uma interacção homem-máquina limitada. A aproximação centrada no cliente elimina ou atenua as duas últimas referidas limitações.

Embora a combinação dessas duas tecnologias (centrada nos servidores e centrada nos clientes) permita a construção de SIW com um razoável nível de qualidade, levantam novas questões em parte devido à existência de distintas tecnologias promovidas por diferentes fabricantes num contexto altamente competitivo. Algumas dessas questões são: Será possível conseguir-se um aceitável nível de interoperação entre os diferentes SIW construídos por diferentes fabricantes? Será possível o desenvolvimento de SIW num ambiente aberto, independente do fabricante? Virá a existir algum standard de interacção com o utilizador (com interfaces baseados em Java-AWT, HTML, capacidades dos *plug-ins*)?

Constata-se que as duas primeiras questões já foram anteriormente questionadas (no princípio da década de 90), no contexto dos sistemas de informação das organizações, e desencadearam como resposta as especificações e arquitecturas de objectos distribuídos (e.g. CORBA) e standards de acesso a bases de dados (e.g. ODBC). Há portanto uma convergência, em termos de problemas comuns e objectivos gerais, entre os sistemas de objectos distribuídos e o que se julga possam vir a tornar-se no futuro os SIW.

7. REFERÊNCIAS

[1] C. Ang, D. Martin. *Constituent Component Interface ++*. Centre for Knowledge Management, University of California, 1995.

[2] J. Baumann, C. Tschudin, J. Vitek, editors. *Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems (Linz, Austria)*. Dpunkt, 1996.

[3] L. Cardelli. Obliq: a Language with distributed scope. *Digital White Paper*, Digital Equipment Corporation, Systems Research Center, 1995.

[4] S. Lucco, O. Sharp, R. Wahbe. Omniware: a Universal Substrate for Web Programming. Fourth International WWW Conference, Boston, In *WWW Journal*. W3C, Dec. 1995.

[5] Microsoft Corporation. *IDC - Internet Database Connector*. http://www.microsoft.com/

[6] Microsoft Corporation. *What is Active-X?* July 1996.http://www.microsoft.com/activex/

[7] Microsoft Corporation. *Visual Basic Scripting Edition*. 1996. http://www.microsoft.com/vbscript/

[8] NCSA, University of Illinois. *Server Side Includes (SSI)*. 1995.

[9] NCSA, University of Illinois. *Common Client Interface Protocol Specification*. 1995.

[10] Netscape Communications. *Netscape Plug-ins*. 1995.

[11] Netscape Communications. *JavaScript Authoring Guide*. 1995.

[12] J. Ousterhout. *Tcl and Tk Toolkit*. Addison-Wesley Publishing, 1994.

[13] D. Robinson, *The WWW Common Gateway Interface Version 1.1*. Internet Draft, 1996.

[14] A. Silva, G. Andrade, J. Delgado. A multimedia database supporting a generic computer based quality management system. In *Proceedings of the 9th ERCIM Database Research Group Workshop*, Darmstadt, Germany,1996.

[15] A. Silva, J. Borbinha, J. Delgado. Organizational management system in an heterogeneous environment - A WWW case study. In *Proceedings of the IFIP working conference on information systems development for decentralized organizations*, Trondheim, Norway, 1995.

[16] J. Spiegel. CORBA Fundamentals and Programming. Wiley, 1996.

[17] J. Borbinha, A. Silva, P. Ribeiro, J. Delgado, P. Caetano. Desenvolvimento de Sistemas e Serviços para a Internet com interface WWW. *Comunicações do 2º Encontro Nacional do Colégio de Engenharia Electrotécnica da Ordem dos Engenheiros*, Dezembro 1995.

[18] World Wide Web Consortium. *Overview of CGI*. 1996. http://www.w3c.org/CGI/

[19] Volant Corp. *htmlScript: HTML Based 4GL for Web Servers*. http://htmlscript.volant.com

[20] Microsoft. *Using Active Server Pages with Microsoft Internet Information Server 3.0*. White Paper. 1996. http://www.microsoft.com/intdev/iis/

[21] Netscape Communications. *The Netscape Server API*. Maio 1998.

[22] S. Genusa, M. Regelski. *Using ISAPI*. QUE, 1997.

[23] G. Cugola, C. Ghezzi, G. Picco, G. Vigna. Analyzing Mobile Code Languages. Em J. Vitek, and C. Tschudin (editores). *Mobile Object Systems – Towards the Programmable Internet*. Lecture Notes in Computer Science, 1222, Springer, Julho 1996.

[24] T. Thorn. Programming Languages for Mobile Code. *ACM Computing Surveys*, 29(3), Setembro 1997.

[25] OMG. *The Common Object Request Broker: Architecture and Specification (CORBA)*. 1991.

[26] IONA Technologies. *OrbixWeb 3 – The Internet ORB*. 1998. http://www.iona.com/products/

[27] ObjectSpace. *Voyager Core Technical User Guide*. Version 1.0. 1997.

[28] ObjectSpace. Part 4: ObjectSpace Voyager CORBA Integration. Em *Voyager Core Technical User Guide*. Version 2.0 Beta 1. 1998.

[29] Inprise Corp. *VisiBroker – Distributed Object Connectivity Software*. 1998.

[30] World Wide Web Consortium. *Jigsaw Overview*. 1998. http://www.w3c.org/Jigsaw/

[31] Sun Microsystems, Inc., JavaSoft. *Java Remote Method Invocation (RMI)*. 1997. http:// www.javasoft.com/products/jdk/rmi

[32] Sun Microsystems, Inc., *Java Server Product Group*, 1998. http://www.javasoft.com/ products/java-server/

[33] Sun Microsystems, Inc., *Java Servlets*, 1998. http://www.javasoft.com/products/java-server/servlets/