6. Part 1 of the Solution:

The Downloadable Architecture

To overcome the shortcomings of C/S architecture, the improved C/S architecture, namely downloadable architecture, is proposed. It allows developers to develop downloadable BISs to be downloaded, installed and run on end-user machines dynamically and automatically from a central location. A BIS architecture shows how a BIS is realized by a collection of elements together with the interactions among these elements (David & Shaw, 1993; Shaw et al., 1995). I discuss these matters in more detail below.

6.1 Elements of the Downloadable Architecture

Like the C/S architecture (Orfali et al., 1999), the downloadable architecture consists of three types of building blocks: required client program, optional application server, and required database. One characteristic in which the downloadable architecture differs from the Windows-based C/S architecture is the automatic deployment of client programs. The client programs of the Windows-based C/S architecture implemented as Windows programs have to be deployed manually. In

contrast, the client programs of the downloadable architecture are implemented as ActiveX-enabled Websites called program warehouses, and will be deployed automatically. Please compare Figure 3 with Figure 4.

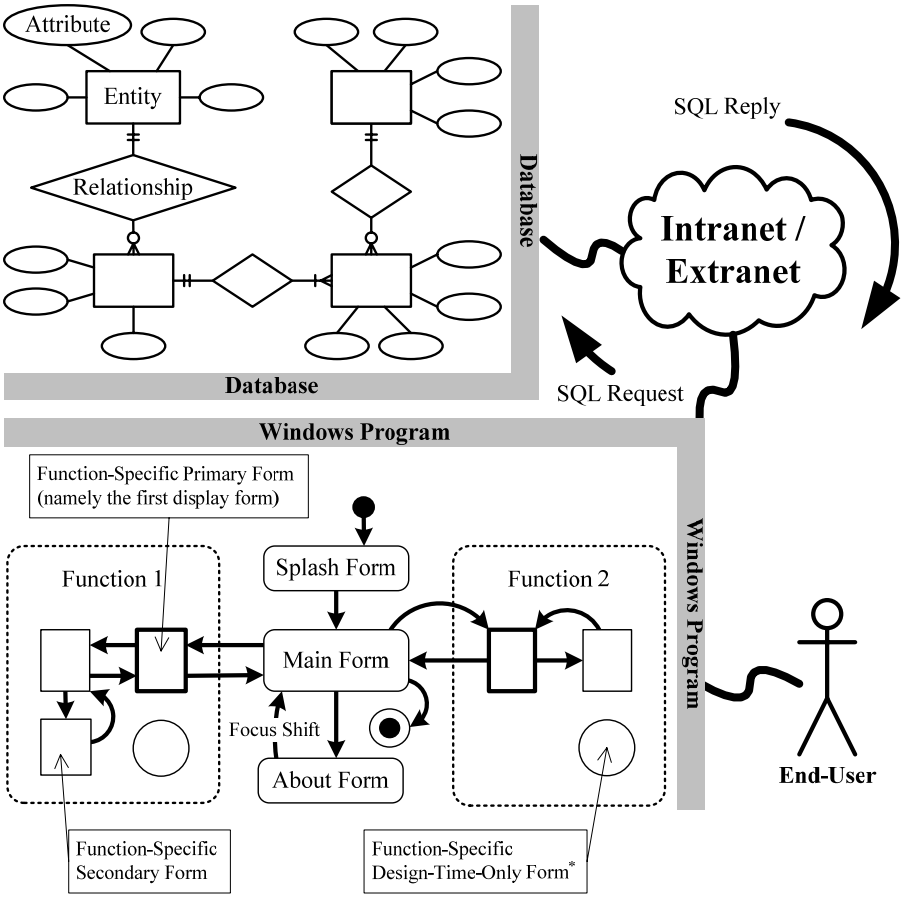
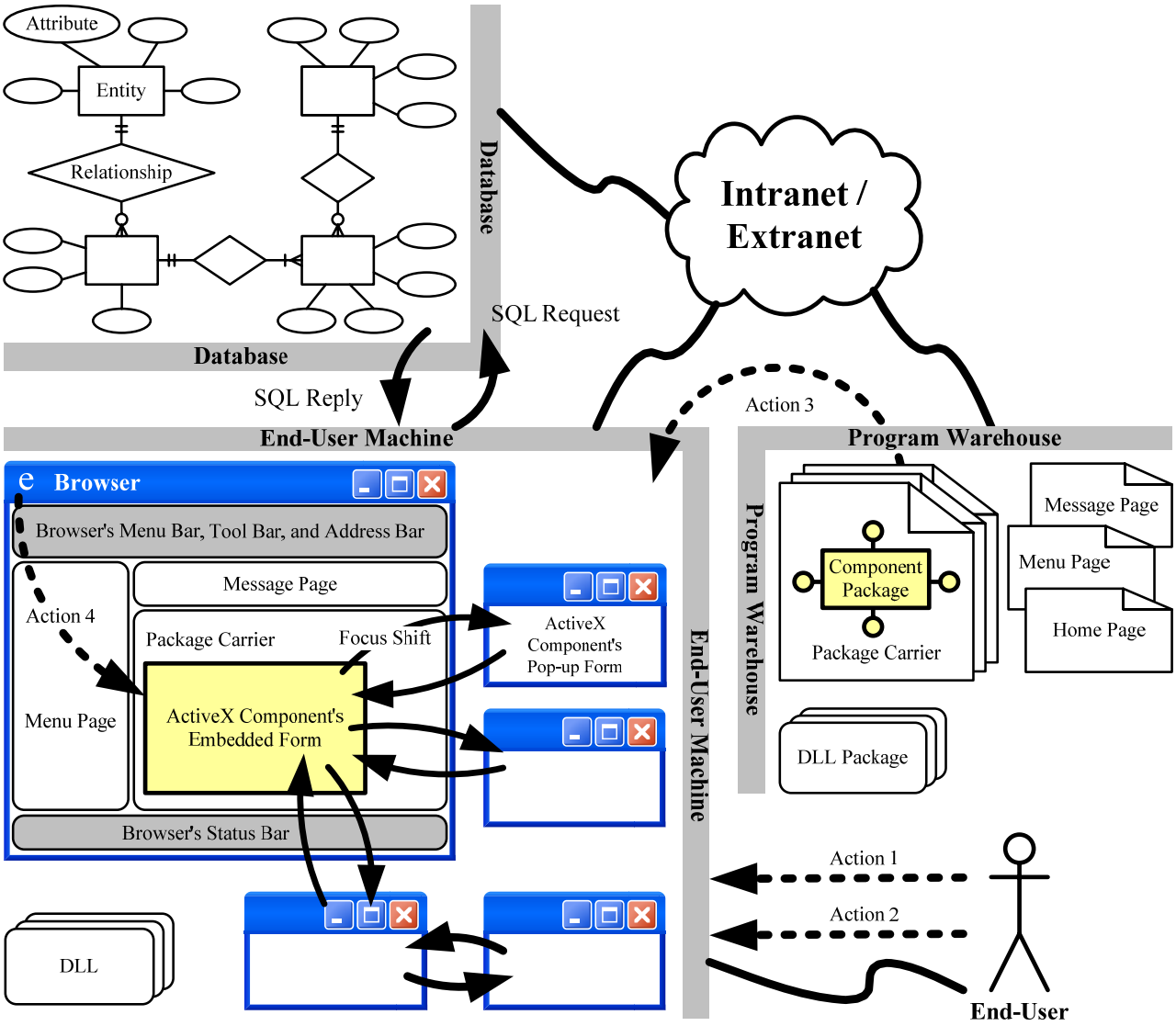


Figure 3. Architectural Overview of Two-Tier Windows-Based C/S BISs



Action 1= End-user opens the home page of the program warehouse.
Action 2= End-user selects a desired function by clicking a menu item on the menu page.
Action 3= Browser checks whether a recent enough version of the function-specific ActiveX component exists in the end-user machine. If it does not exist, the component package and related DLL packages be downloaded, extracted, and installed.
Action 4= Browser creates and initializes a ActiveX component instance.

Note that the ActiveX component's design-time-only forms are not shown in this figure.

Figure 4. Architectural Overview of Two-Tier Downloadable BISs

A program warehouse consists of a home page, a menu page, a message page, one or more package carriers, one or more component packages, and zero or more DLL packages. The home page, the menu page, the message page, and the package carrier all are simple Web pages. The home page is used to inspect the end-user environment, customize the Website for the end-user, split up the browser window into frames, and other such Website initialization processes. The menu page is used to list all available functions of the client program. The message page is used to display information about the system. The package carrier is used to deliver a component package and zero or more DLL packages to the end-user machine. In addition, the package carrier also performs some user interface processes to increase the usability of the client program. The component package and the DLL package both are CAB-format compressed files (CAB files) that include one or more downloadable files. The component package contains an ActiveX component (OCX file) and its installation instructions (INF file), but the DLL package contains only an ActiveX component's dependent DLL. Each of the ActiveX components is composed of an embedded form, zero or more pop-up forms, and zero or more design-time-only forms, to implement a function of the client program that appears as a menu item on the menu page. Figures 11-17 contain examples of downloadable BISs.

6.2 Interactions among Elements

A downloadable BIS is a Web-based system, and can be accessed through an ActiveX-enabled Web browser from anywhere with Intranet or Extranet connectivity. Here, I outline interactions among elements (see Figure 4).

Firstly, an end-user visits the program warehouse, and then the menu page is displayed within the end-user's browser.

Secondly, the end-user selects the desired function from the menu page, and then a function-specific package carrier is downloaded to the end-user machine.

Thirdly, the browser analyzes the package carrier to retrieve the ActiveX component's globally unique identifier (GUID), the ActiveX component's minimum required version, and the component package's download location. Using this information, the browser checks whether a recent enough version of the ActiveX component exists in the end-user machine. If it does not already exist, the browser downloads the component package, extracts the included ActiveX component and ActiveX component's installation instructions, and then installs the ActiveX component against the installation instructions.

Fourthly, the browser parses ActiveX component's installation

instructions to retrieve each dependent DLL's file name, each dependent DLL's minimum required version and each dependent DLL package's download location. Using this information, the browser checks whether a recent enough version of the DLL exists in the end-user machine. If it does not already exist, the browser downloads the DLL package, extracts the included DLL, and then installs the DLL against the ActiveX component's installation instructions.

Finally, the browser instantiates and initializes the ActiveX component before an ActiveX component instance, which embeds in the package carrier, is displayed within the browser.

6.3 Communications among Elements

The downloadable architecture is loosely coupled, meaning one element can be understood without examining the other, and one element often can be changed without changing the other. Nevertheless, communication among the elements of downloadable architecture is unavoidable. In downloadable architecture, there are five types of communication: communication between ActiveX component instances, communication between AtiveX component instances and Web scripts, communication between client-side Web scripts and server-side Web scripts,

communication between client-side Web scripts, and communication between server-side Web scripts. The inter-module communication methods provided by high-level programming languages, such as global variable, procedure call, and message passing (Dershem & Jipping, 1995; Sebesta, 2007), cannot work in the downloadable architecture (note that downloadable system is created using multi-languages and is a multi-process system). The interprocess communication methods provided by operating systems, such as IPC, RPC, DCOM, and CORBA (Microsoft, 2007c; OMG, 2001; Silberschatz et al., 2004), also cannot work in the downloadable architecture (note that there is no overlap between lifetimes of ActiveX component instances). Moreover, most of the session state maintenance methods provided by Web technologies, such as ASP/PHP session variable, URL-encoded variable, hidden form variable (Kristol, 2001; Wang & Katila, 2003), are partial solutions. In fact, session cookie (Kristol, 2001; Wang & Katila, 2003) is the only solution that supports all such types of communication. Table 2 lists the frequently used Win32 API functions that include cookie functions and hyperlink navigation functions.

6.4 Web Browser Security Settings

For security reasons, by default, most Web browsers will block or warn against the download and installation of ActiveX components. Thus,

it is necessary to configure the browser's security settings to enable ActiveX components. Taking IE 6.0 as an example, follow the steps below to configure the browser's security settings.

1. To assign the program warehouse to the Trusted sites zone:

   (1) In IE, on the Tools menu, click Internet Options.
   (2) On the Security tab, click the Trusted Sites icon, and then click Sites.
   (3) Clear the Require server verification (https:) for all sites in this zone check box.
   (4) Under Add this Web site to the zone, type the URL of the program warehouse.
   (5) Click Add, and then click OK twice.

2. To configure the security level for the Trusted sites zone:

   (1) In IE, on the Tools menu, click Internet Options.
   (2) On the Security tab, click the Trusted Sites icon, and then click Custom Level.
   (3) Scroll down to ActiveX controls and plug-ins, and then set all options to Enable.
   (4) Click OK twice.

Table 2. Frequently Used Win32 API Functions

Function	Description
InternetGetCookie	Retrieves the cookie for the specified URL
InternetSetCookie	Creates a cookie associated with the specified URL
HlinkGoBack	Executes a hyperlink jump backward within the navigation stack
HlinkGoForward	Executes a hyperlink jump forward within the navigation stack
HlinkSimpleNavigateToString	Executes a hyperlink jump to a new document or object
HlinkSimpleNavigateToMoniker	Executes a hyperlink jump, specified by a moniker, to a new document or object