Design of a Universal Web Application Installer

Draft Version 1.0

By Brad Touesnard

CS4983 – Senior Report

Supervised by Prof. Andrew McAllister

Fredericton, New Brunswick

17 March 2005

UNIVERSITY OF NEW BRUNSWICK

FACULTY OF COMPUTER SCIENCE

Executive Summary

Originally web applications were built by developers and usually installed by developers or system administrators. Today however, there is an increasing number of users installing web applications with limited skills and knowledge. These include users wanting to publish a web log or setup a bulletin board system. There are currently thousands of open-source web applications available, but no standardized, simple way for them to be installed. Many of the more popular web applications come bundled with their own install program but users are still obliged to unpack the files from an archive and visit a web address to run the install program. This may sound simple to web developers, but to the average user it does not. Additionally, the install program almost always lacks the privileges to perform all of the operations to completely install the web application. Often times, the install program asks the user to change permissions on certain files and folders. Not only is this an irritating request, but for many users it is beyond their knowledge. The user should not be expected to perform additional installation tasks after running the installer. That is, the installer should perform all of the operations necessary to fully install the web application.

The challenge is designing a web application installer that will feature all the possible operations needed to successfully install any web application, from start to finish.

Table of Contents

1.0 Objective

The objective of this report is to present a design for a standardized and user-friendly installer program that will enable easy installation of web applications by users of all skill levels.

2.0 Scope

A web application (WA) is defined as a software application delivered to end-users from a web server over a network such as the World Wide Web (WWW) or an intranet. Common examples of WAs include webmail, message boards, and web logs (blogs). [1], [2]

Java Applets and Java Scripts are sometimes referred to as WAs but do not fit the definition of a WA described above. These are browser-based applications and do not require a web server to be delivered. It is important to distinguish between these two types of applications and note that this report will not be discussing browser-based applications.

3.0 Definition of Terms

Universal	The term “Universal” in Universal Web Application Installer (UWI) simply refers to the capacity of the installer program to install any WA despite its implementation language.
Web Application	An application delivered to end-users from a web server over a network such as the World Wide Web or an intranet. [1]
Web Server	**A** software program that runs as a service on a computer that is responsible for serving web content.
Web Hosting	A service that provides users with online systems for storing information, images, video, or any content accessible via the web. [3]
Shared Web Hosting	One of the most common types of web hosting. Distinguished by many web sites sharing the same web server. The server is usually managed by a root user and other users on the server have limited access to server resources.
Web Hosting Control Panel	A web application enabling web hosting clients to manage their email addresses, FTP accounts, domain names, and other account features.
shall	In the functional requirements section, “shall” implies that functionality is required to be implemented.
should	In the functional requirements section, “should” refers to the functionality that is highly recommended for implementation but is not required.

4.0 The Current State of Web Application Installation

A common way to install a WA today is by following directions provided in the WA's documentation. Many WAs include an installer program that eases some of the installation tasks. However, there is still a lot of manual work that must be done before and after using an included installer.

First, the user must download the WA package, unpack it, and follow any directions for setting up the installer. Often a configuration file must be modified or created before the installer can begin its work.

After the installer has completed its work, there are often additional tasks that must be performed to complete the setup process. Most often, the permissions on directories must be changed so that the web server has access to write files to those directories. A "thumbnails" directory is a typical example of a directory that would need to be writeable so that images that have been resized can be saved.

One of the biggest problems with the installers included with WAs is that they do not have access to perform all of the operations they need to perform to completely setup the WA. An easy solution that may come to mind would be to simply give the WA access to perform the needed operations. Unfortunately, giving the WA this level of access would pose an unacceptable security risk and is not usually considered an option.

Recognizing this problem, web hosting control panel (CP) developers have decided to integrate their own installers into their CPs because the CP software already has access to perform all the needed operations to fully install a web application. cPanel is currently the industry leading CP for the Linux operating system and offers sixteen

“Pre-Installed” web applications. [4]   In addition, Fantastico is a very popular extension to cPanel offering over forty additional web applications to install. [5]

Although cPanel and Fantastico are valued by their users and web hosting professionals, they are both proprietary software packages and only allow installation of the web applications they have chosen. Therefore, if there is a web application not currently available from the cPanel or Fantastico installers, the user has no choice but to manually install the web application.

The need for an open-standard web application installer enabling the complete installation of any web application is clear. An open-standard web application installer would benefit web application developers, web hosting companies, and most importantly the end-user.

5.0 Universal Web Application Installer Design

5.1 Architecture

In order to begin understanding the details of the UWI system, it is important to first establish a picture of the overall structure of the system, all its major components, and how they work together (Figure 1).

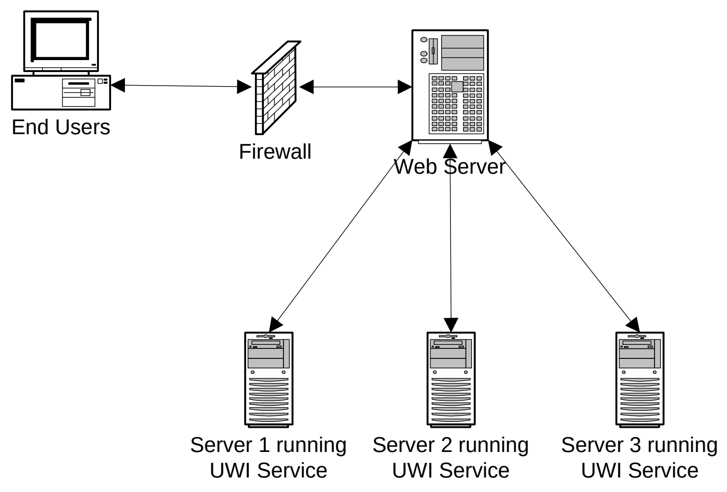

End Users
Firewall
Web Server
Server 1 running UWI Service
Server 2 running UWI Service
Server 3 running UWI Service

Figure 1: Network diagram of the UWI system in a multi-server environment

The UWI system is composed of two main components: the UWI interface operated by end-users and the UWI service running on the user's server. In a single-server environment, the UWI interface and service would reside on the same server.

Operation of the UWI system is based largely on input from end-users. It is the end-user who dictates via the web-based interface the operations performed by the service running on their server. The UWI service runs on all servers in which the user has access to install a WA. Scenario 1 better illustrates the communication between the UWI interface and the UWI service. There are some details such as user authentication (security) that have been left out of Scenario 1 to keep it simple.

Scenario 1: Installing Forum Software

John has a web hosting account on Server A and wishes to install phpBB, a popular forum software WA. To accomplish this, John points his web browser to the address of the UWI interface, chooses phpBB from the list of WAs to install, and then submits his installation request. The request is received by the UWI service running on Server A which proceeds to install phpBB as requested. After installation has completed, the UWI service sends a message back to the UWI interface informing John that installation has succeeded or failed.

5.2 Functional Requirements

5.2.1 The Interface

The UWI interface shall enable the user to login using a unique username and password. Once logged-in, the interface shall enable the user to choose a WA package to install that they have saved to either their web hosting account or local hard drive. The interface should allow the user to browse the list of files and directories in their account and select the WA package. The interface should also enable the user to choose from a list of available WAs to install. The WA list is an optional feature that can be setup by an administrator of the UWI system. To setup this list, the administrator must create a *repository* of WAs. The repository is simply a directory containing WA packages and a document (XML is recommended) describing the packages within this directory. When a user selects a WA from the list, the interface shall display a button to begin a new install and list any previous installations including the date and location of installation.

When a new install is initiated by the user, the interface shall send a request to the UWI service to begin installation of the selected WA. Next, the interface shall display input forms and instructions generated from information in the response from the UWI service. These input forms will collect information from the user essential to completing installation of this particular WA. Once this information has been collected from the user, the interface will ask the user where in their account to install the WA. The interface should give a recommended location by default but also shall enable the user to browse the list of directories in their account and select a directory in which to install the WA. The interface will also enable the user to create a new directory if they wish to do so. If a the WA requires a database, the interface shall ask the user to input a database server address, a username and password, a database name, and an optional prefix for data table names. The interface should also enable the user to test their connection to the database.

Now that all of the information has been collected to execute complete installation, the interface shall display a summary of all of the information collected from the user and enable the user to return to modify information previously entered or execute installation. When the user executes installation, the interface must send a request to the UWI service to install the WA containing all of the information collected from the user. Next, the interface shall display the response received from the UWI service detailing the success or failure of installation.

5.2.2 The Service

The UWI service shall authenticate an incoming connection by checking the incoming username against the usernames allowed to access the server. If a matching username is found, the UWI service shall check the incoming password against the password associated with the matching username.

When a new install is requested from the interface, the UWI service shall unpack the requested WA package to a temporary location, read an install script included with the package, and return instructions to the interface dictating information to display and collect from the user. Further information about the install script is presented in Section 5.3.

When a request to execute installation is received from the interface, the service shall use information received from the interface and instructions read from the install script to perform operations on the unpacked WA files. Operations include modifying file and directory permissions, adding and removing files and directories, modifying and overwriting files, and copying and moving files and directories **within the unpacked WA directory (see Section 5.6)**.

Once all operations have been performed, the service shall move files to the install location specified by the interface. Next, the service shall return a message to the interface describing the installation success or detailing any failures that occurred during the installation procedure.

5.3 The Install Script

The purpose of the install script is to describe the WA and the installation processes that must be executed for a complete installation. An XML-based install script would enable easier reading and writing of the installation procedures for developers and is highly recommended as the document format. In addition to easy reading and writing, an XML-based install script enables embedding of XHTML allowing the script's author to write the installer controls in valid XHTML as they would write the controls for any web page. The following is a brief specification of the UWI XML installer script.

Every UWI installer script must begin with the <uwi> element with a mandatory *version* attribute specifying the version of UWI installer script to which the document conforms. The current specification is version 0.1.

The sub-element of <uwi> is a single <app> element which contains information about the application and how to install it.

Sub-elements of <app>:

Element	Description
title	The title of the web application
author	The person, group or organization who wrote the web application
description	A phrase or sentence describing the web application
link	The URL of the web application’s official web site
version	The version of the web application
configuration	Contains elements related to configuration of the installation
operations	Contains elements describing installation operations to be executed

Sub-elements of <configuration>:

Element	Description
database	Contains elements concerning database configuration
steps	Contains <step> elements for each installation step specific to this web application

Sub-elements of <operations>:

Element	Description
chmod	Change the specified file or directory permission. Has attribute *mode* that accepts numeric Unix file permission modes. e.g. 0777
move	Move the specified file or directory (including its contents).
remove	Remove the specified file or directory (including its contents).
copy	Copy the specified file or directory (including its contents).
mkdir	Create a new directory.
mkfile	Create a new file.
sql	Execute the specified query.

Any {VAR_VARNAME} strings found in the contents of <operations> sub-elements are replaced by the value of VARNAME which was defined by the user during configuration.

Sub-elements of <database>:

Element	Description
required	Specifies whether a database selection is required
dbms	Contains elements describing a database system

Sub-elements of <step>:

Element	Description
any element of XHTML 1.0	Any valid XHTML 1.0 elements can be inserted here and will be used to display the given step input forms

Sub-elements of <dbms>:

Element	Description
title	A description of the database system
script	A path to the database script to execute for installation

Sub-elements of <mkfile>:

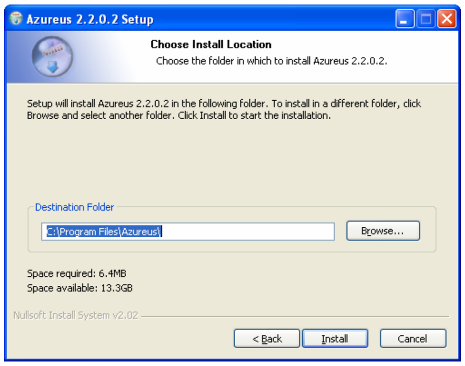
Element	Description
name	The name and path of the file to be created
data	The contents of the file

The RSS 2.0 Specification was used as a guide for this brief specification of the UWI Installer Script. [6] An example of an XML-based install script is presented in Appendix A.

5.4 User Interface Design

One of the keys to the success of this project is a professional user interface design with which the user is familiar. As the Windows operating system is currently dominating the operating system market, the majority of end-users are familiar with

Windows software installers. Common Windows installers include Microsoft's Scriptable Installer, Wise, InstallShield, and the open-source NullSoft Scriptable Install System (NSIS); all of which have basically the same interface design. [7]


Azureus 2.2.0.2 Setup

Choose Install Location

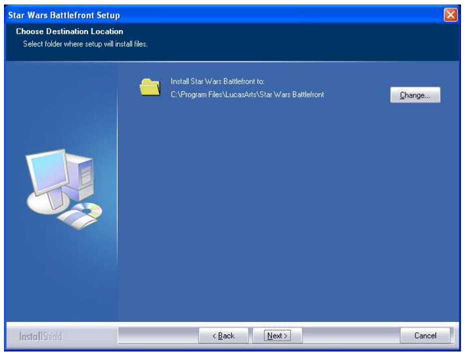
Choose the folder in which to install Azureus 2.2.0.2.

Setup will install Azureus 2.2.0.2 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files\Azureus\

Space required: 6.4MB
Space available: 13.3GB

Nullsoft Install System v2.02


Star Wars Battlefront Setup

Choose Destination Location

Select folder where setup will install files.

Install Star Wars Battlefront to:
C:\Program Files\LucasArts\Star Wars Battlefront

Figure 2: NullSoft Scriptable Install System (left) and InstallShield (right)

The UWI user interface design should employ the same layout design as these popular Windows installers and should be similar in overall appearance. A title bar should occupy the entire top portion of the screen and should display the title and a short description of the current step in the install process. Detailed instructions about the current step should be placed directly under the title bar followed by the controls for the current step. Controls to move to the next step and back to the previous step should be located at the bottom of the screen. A button to cancel the installation should be located in the bottom right of the screen. The basic layout of the UWI interface is shown in Figure 3.

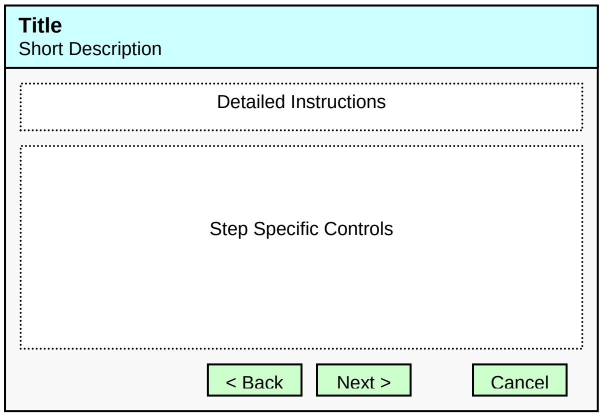


Figure 3: Basic Layout Design for the UWI Interface

5.5 Implementation Language, Platform and Protocol

PHP is a web development language that presents many advantages over other languages for use in this project including support for most major operating systems and most major relational database systems. Since PHP supports multiple operating systems and database systems, the UWI system (both the interface and service) implemented in PHP can also offer this support. This is particularly important because the more server configurations that are supported by the UWI system, the more WAs can make use of the installer and the better chance that it will be available to end-users.

The protocol used by the UWI system should be one which is simple to understand and widely supported in the industry. XML-RPC is a mature protocol which enables execution of procedures on another server by sending a request as an XML document. There are currently several libraries available in PHP that implement the XML-RPC protocol which could be used for this project. The XML-RPC library from usefulinc.com has been used in the past and is highly recommended. [8]

5.6 Security

There are several security vulnerabilities that are common in WA currently running on the web today including SQL injection and cross-site scripting. Vulnerabilities that are specific to WAs developed in PHP include register_globals insecurities and file includes. [9], [2] The details of these vulnerabilities and how to prevent them are outside the scope of this report but it is important for developers to be aware of these issues when developing the UWI system. Recommended readings include the DevShed article “PHP Security Mistakes” [10], the ONLamp article “PHP Security” [11] and the “Security” chapter in “Profession PHP4” [9].

As mentioned in Section 5.1, the UWI system can be setup in a multi-server environment as well as a single-server environment. The single-server environment presents significantly stronger security as all communication between the UWI interface and the UWI service is done on the server and does not go out over the network. In a multi-server environment, communication between the interface and service should be encrypted using SSL. The open-source cURL and OpenSSL packages enable encryption of communication with PHP.

To accept connections from the UWI interface, the UWI service must open a port on the server on which it is running. Opening a port on a publicly accessible server is an invitation for attacks. To prevent the possibility of an attack, security can be implemented at the network level by installing a firewall between the server running the UWI service and the public internet. The firewall will only allow traffic on specified ports to reach the server running the UWI service. Therefore, by disallowing traffic for the port on which the UWI service is listening, we can minimize this vulnerability.

To further secure the UWI service, we can accept incoming connections and verify their IP address against a list of allowed IP addresses. Only incoming connections with allowed IP addresses will be accepted, the rest will be disconnected. It is important to note that there are two forms of IP addresses: the common format is four sets of three digit numbers separated by periods (e.g. 207.142.131.236) and the less common is a straight-number format (e.g. 3482223596). Knowing that these formats are equally valid when writing an IP address filter is essential to security. A known hacking technique is to use the straight-number format to bypass filters checking only for the period-separated format.

Since the UWI service will be manipulating files on the server on which it is running based on instructions from the end-user and the install script, it is crucial to the security of the server that operations are limited to files within the temporary directory of the unpacked WA. If operations are not limited to the WA files, an install script could be tailored to carry out harmful acts on any of the user's files. For example, an install script could be written to delete files in the user's directory.

To ensure that operations are only executed on the WA files, the path to the temporary WA directory should be appended to every file path. For example, if the temporary WA directory was */home/brad/temp/phpBB2/* and an operation is to be carried out on the file *include/config.php*, then the operation would be executed on the path */home/brad/temp/phpBB2/include/config.php*. It is also important to recognize that file and directory paths can be manipulated by using relative paths and symbolic links (on Unix based systems). For example, a relative path like *../../config.php* could be given which would translate to the path */home/brad/config.php*, outside the temporary WA directory. To prevent such manipulation, the PHP function *realpath()* can be used to translate the path, then the translated path can be checked if it is within the temporary WA directory.

6.0 Conclusion

In order to gain immediate and widespread acceptance, the designed UWI system will behave and appear much like the pseudo-standard design presented by Windows-based installers. The interface design will present similar features because these features are already accepted by users of Windows installers. The implementation language will offer support for most operating systems and most database systems. The UWI system will be implemented with security as a high priority to ensure that system administrators will not hesitate to install the UWI system and offer it to their end-users due to security concerns.

It is clear that a standard WA installer is much needed by end-users to facilitate WA setup and eliminate the need for manual installation. The UWI design outlined in this report has addresses issues with current installation methods and offered a comprehensive solution to this problem.

7.0 Recommendations

It is highly recommended that this project be developed and released under the General Public License (GPL) or another suitable open-source license. Releasing the UWI system as open-source will enable quick adoption as an unofficial industry standard WA installer and will increase the security of the system. I recommend "The Cathedral and the Bazaar" by Eric S. Raymond [12] for more details on the advantages of open-source software.

The described UWI design currently does not allow for upgrading an installed WA. Upgrading is an important part of the application maintenance as security updates are often included as upgrades. Upgrading often presents a similar set of operations as installing and including the upgrading feature into the UWI design would require little effort. Although, the biggest problem with upgrading is that any modifications that have been made to the WA files to be overwritten will be lost. A sophisticated merging feature like the one included with most source control systems (Microsoft Visual Source Safe, CVS, Subversion, etc.) would enable the end-user to update the WA files without losing their modifications. In addition to upgrading, the ability to uninstall an installed WA is also a feature that would be useful.

In order to reach as many end-users as possible, the UWI system should employ other features common to installers such as the NullSoft Scriptable Install System. For example, the UWI system could be developed with multiple language support enabling additional languages to be added as needed. [13]

Bibliography

1. Unknown, “Web Application,” *Wikipedia*, http://en.wikipedia.org/wiki/Web_application [accessed 2005-03-10]

2. Unknown, “Building and Deploying Secure Web Applications,” London: Info-Tech Research Group, 2002.

3. Unknown, “Web Hosting,” *Wikipedia*, http://en.wikipedia.org/wiki/Web_hosting [accessed 2005-03-10]

4. Unknown, “cPanel Features,” *cPanel.com*, http://www.cpanel.net/features-cpanel.html [accessed 2005-03-12]

5. Unknown, “Fantastico De Luxe,” *netenberg.com*, http://www.netenberg.com/fantastico.php [accessed 2005-03-12]

6. Rogers Cadenhead, Adam Curry, Steve Zellers, “RSS 2.0 Specification,” *RSS Advisory Board*, http://blogs.law.harvard.edu/tech/rss [accessed 2005-03-15]

7. Unknown, “Installer,” *Wikipedia*, http://en.wikipedia.org/wiki/Installer [accessed 2005-03-13]

8. Unknown, “An XML-RPC client and server for PHP,” *usefulinc.com: XML-RPC*, http://xmlrpc.usefulinc.com/php.html [accessed 2005-03-16]

9. Luis Argerich et al, “Professional PHP4,” Apress, 2003.

10. Dave Clark, “PHP Security Mistakes,” *DevShed*, http://www.devshed.com/c/a/PHP/PHP-Security-Mistakes/ [accessed 2005-03-15]

11. John Coggeshall, “PHP Security,” *ONLamp.com*, http://www.onlamp.com/pub/a/php/2003/07/31/php_foundations.html [accessed 2005-03-16]

12. Eric S. Raymond, “The Cathedral and the Bazaar,” http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ [accessed 2005-03-16]

13. Joost Verberg, “Features,” *NullSoft Scriptable Install System*, http://nsis.sourceforge.net/features/ [accessed 2005-03-13]

Appendix A: An XML-based Installer Script Example

```
<uwi version=”0.1”>
  <app>
    <title>phpBB2</title>
    <author>phpBB Development Group</author>
    <description>
      A popular open-sourced bulletin board web application.
    </description>
    <link>http://www.phpbb.com</link>
    <version>1.1</version>
    <configuration>
      <database>
        <required>true</required>
        <dbms id=”1”>
          <title>MySQL 3.2.x</title>
          <script>db/schemas/mysql.sql</script>
        </dbms>
        <dbms id=”2”>
          <title>PostgreSQL 7.x</title>
          <script>db/schemas/postgre.sql</script>
        </dbms>
        <dbms id=”3”>
          <title>Microsoft SQL Server 7/2000</title>
          <script>db/schemas/mssql.sql</script>
        </dbms>
        <dbms id=”4”>
          <title>Microsoft Access (via ODBC)</title>
          <script>db/schemas/msaccess.sql</script>
        </dbms>
      </database>
      <steps>
        <step>
          <fieldset>
            <legend>phpBB Configuration</legend>
            <label for=”sname”>Site Name</label>
            <input type=”textbox” name=”sname” size=”20” id=”sname” />
          </fieldset>
        </step>
      </steps>
    </configuration>
    <operations>
      <sql>
        UPDATE {VAR_DBPREFIX}config SET config_value = ‘{VAR_SNAME}’
        WHERE config_name = ‘sitename’
      </sql>
      <remove>install.php</remove>
      <remove>upgrade.php</remove>
      <remove>db/schemas</remove>
      <chmod mode=”0777”>images/avatars</chmod>
    </operations>
  </app>
</uwi>
```