WGDE -- WAP Game Development Environment

Yet-Shiang Wang, I-Chen Wu, Wen-Nung Tsai
Department of Computer Science and Information Engineering,
National Chiao-Tung University
{*wangys, icwu, tsaiwn*}*@csie.nctu.edu.tw*

Abstract-*Due to the abundant resources and diversified information services, the popularity of Internet has rapidly grown recently. However, in the traditional Internet, user can only access services at some fixed locations. Because the progress of technology in wireless communications and the portable communication devices, along with the wireless application protocol, users can access Internet information through the cellular phones that support Wireless Application Protocol (WAP). Many wireless applications had been developed, including stock application, restaurants services and on-line games, etc. Among these applications, the on-line wireless games will become more popular in the near future.*

WAP has been criticized as "Wait And Pay" due to lack of killer applications. This is because that the wireless terminals and communications have many constraints. It is not an easy task to develop WAP games, especially a large one. In this paper, we design and implement a platform that provides useful classes for WAP game development. The platform provides a Wireless Markup Language (WML) page generator, called the WAP Game Foundation Classes (WGFC), containing layout manager, database access, and game templates. Programmers can develop WAP games easily without much knowledge of the WML. Moreover, the WGFC also deals with common security issues, including authentication, integrity, and confidentiality.

Keywords: servlet, WAP, WML.

1. Introduction

Since the 802.11x wireless networks mature, we can access the Internet services anytime and anywhere via wireless devices, like mobile phones. Mobile phones had become more and more popular, because they are small, light-weight and low-cost. Now, the mobile phones are not just mobile phones; they can be applied to many value-added services.

Before June in 1997, Ericsson, Motorola, and Nokia specified their own wireless application protocols that were not interoperable. The development of wireless applications was greatly limited. Open standards can make entire industry grow well. In order to prevent incompatibility and the warfare of the standards, Ericsson, Motorola, Nokia and Phone.com founded the Wireless Application Protocol Forum in 1997 [14]. The WAP forum is an industry group dedicated to the goal of enabling sophisticated telephony and information services on handheld wireless devices. These devices include mobile telephones, pagers, personal digital assistants (PDAs) and other wireless terminals. Recognizing the value and utility of the World Wide Web architecture, the WAP Forum has chosen to align its technology closely with the Internet and the Web.

Without worrying about the compatibility, today everyone can provide any equipments or services that follow the WAP specification. Such characteristic brings new ideas of services and opens new markets. Subscribers can access traditional WWW information by WAP-enabled terminals, and are no longer limited at fixed places. Whether service providers are using D-AMPS, GSM, GPRS, CDMA, or UMTS, users can get services and information transparently.

The WAP services are mainly developed for the mobile network. The most proper services are those personal, instant, necessary, and non-video services, such as e-commerce. Content providers or service providers offer information services through the WAP Gateway. Due to the small, colorless displays, the wireless terminals can only support WML decks instead of HTML pages. There are many WAP applications, including the Internet information access, business efficiency promotion, notification services, e-commerce, advanced telecommunications services, and entertainment uses.

The market research by Datamonitor claimed that in 2005 there would be 200 million players competing head-to-head in games built in mobile phone in America and the west Europe. It is almost 80% of the mobile phone population, and it would be a 1.6 billon dollars market! In Asia, there would be at least one billon dollars wireless game market in the year of 2005.

1.1. Wireless Application Protocol

The Wireless Application Protocol (WAP) is an open, global standard that empowers mobile users with wireless devices to easily access and interact with information and services instantly.

WAP is designed to work with most wireless networks such as CDPD, CDMA, GSM, PDC, PHS, TDMA, FLEX, ReFLEX, iDEN, TETRA, DECT, DataTAC, Mobitex, etc. The WAP creates a new

markup language called Wireless Markup Language (WML), which is different from Hyper Text Markup Language (HTML) because of the narrowband network connection. Thus, WAP introduces a gateway between the terminals and the servers, WAP gateway. The WAP gateway translates received requests so WAP-enabled terminals can browse normal HTTP pages. Although the data transmission rate of current mobile terminals is 9.6Kbps, far slower than the 56Kbps of home modems, the new generation technology GPRS will rise the rate to 110Kbps. It will be more convenience to access the Internet.

1.2. Wireless Markup Language

WML is a markup language based on XML and is intended for use in specifying content and user interface for narrowband devices. It has four major functional areas [14]:

- **Text presentation and layout.** WML includes text and image support, including a variety of formatting and layout commands. For example, boldfaced text may be specified.
- **Deck/card organizational metaphor.** All information in WML is organized into a collection of cards and decks. Cards specify one or more units of user interaction (e.g., a choice menu, a screen of text or a text entry field). The size of one deck is limited to 1.4K bytes.
- **Inter-card navigation and linking.** WML includes support for explicitly managing the navigation between cards and decks. WML also includes provisions for event handling in the device, which may be used for navigational purposes or to execute scripts.
- **String parameterization and state management.** All WML decks can be parameterized using a state model. Variables can be used in the place of strings and are substituted at run-time.

WML is an XML language and inherits the XML document character set. Figure 1 shows the WML architecture.

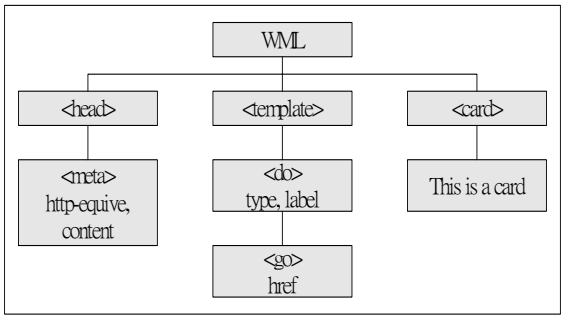

Figure 1. The WML architecture.

1.3. Related Work

Lilja [6] discussed the abilities of WAP devices widely. They suggested that one should take care more about the capabilities of different WAP-aware terminals, like mobile phone, PDAs, and PWAs.

Cannatard and Pascum [7] suggested an object-based architecture to build the WAP applications. They also presented a session manager to against unexpected disconnections which any wireless applications must deal with.

Metter and Colomb [8] investigated problems associated with the conversion process by examining the conversion of a functional information system. They focused on the proper layout of the documents after conversion.

Goeschka and Schranz [9] developed an object-oriented web constructional tool that builds web servers with UML and Jessica. They discussed the constraints on the pure HTTP and use database to build large scaled applications. They also emphasized the separation of the layout elements and programming logic.

Ghosh and Swaminatha [10] studied security problems in mobile e-commerce. Encrypted communication protocols are necessary to provide confidentiality, integrity, and authentication services for m-commerce applications.

BEA [12] has designed the htmlKona which simplifies the task of programmatically generating complex HTML documents. Using htmlKona, programmers code the web pages in a object-oriented way, for every HTML tags are formatted with objects. The same idea is also applied to WML documents. The WebLogic Server has built-in package weblogic.apache.wml which treats WML tags as elements. This tool helps generate web pages, yet other problems remain in WAP game development.

Although there have been many studies for HTTP applications, few researches have been done on wireless game development, especially on WAP games [2][3][4][5][8][11]. In section 2, we will discuss general problems that programmers must handle.

2. The WAP Game Design Issues

A wireless game is restricted by the computing ability of terminals and the quality of wireless network connections. The WAP forum suggests that it is better to build a WAP application in the client-server model: they are WAP terminals, WAP gateways in the middle way, and application servers. The WAP programming model is shown in Figure 2.

To deploy wireless games, it is not sufficient to simply guarantee access to data. We also must consider the constraints on wireless communications before starting to design the wireless game platform. These constraints are described as follows [7]. The constraints on terminals are:

- **Small, low resolution displays and limited user-input facilities.** Displays have few lines of text and low resolution. Few handheld terminals have mouse, and they even have few keys. Some devices have touch screens and voice menu. Hence, different user interface is required.
- **Limited computational resources (CPU, RAM).** Terminals usually have slower CPU and smaller memory size in order to extend the battery life, to lower heat and costs.
- **Cookie technique is not available**. Terminals that support WAP still do not support Cookie technique until now. If game programmers want to bypass players' state, they must do something else such as using database to store players' data.

The constraints on communication are:

- **Low bandwidth, high latency, and unsteady connection.** The value of wireless networks bandwidths currently range from 0.3 to 110Kbps, some order of magnitude lower than wired ones. The main causes of unsteady connection are the blank out period during handover.
- **High Transmission Bit Error Rate.** Error rates are greater than in wired network, and it is more difficult to guarantee Quality of Service.
- **Low predictable service availability.** Wireless networks can suffer short or long periods of inaccessibility, due to congestions or faults.

The constraints on applications are:

- **WAP, as HTTP, is stateless nature.** TCP can obtain connecting state between communications. However, HTTP and WAP are stateless protocols and do not keep any user sessions.
- **Servers do almost all the computation.** Due to the limited computational resources of the terminals, the servers do almost all the computation. Usually, the terminals just display the results received from the servers.

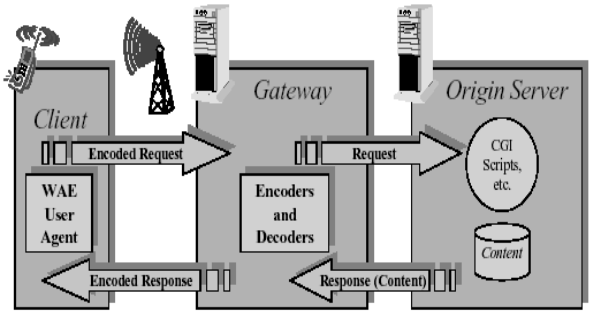

Figure 2. The WAP programming model.

- **The web pages could be hacked easily.** Programmers must ensure any requests are validate, especially from distrusted clients. The web pages could be saved and modified easily on PCs. As a result, it is necessary to use some mechanisms to check the integrity of the pages.

If one wants to develop WAP games, he or she will face with these constraints. The parts of the terminals and communication are not solvable by the software technologies. However, there exist some methods, which we have shown in the section of related work, to make them for the needs of game applications.

Generally speaking, a game with the following properties can be implemented on WAP more easily than others. First, the game rule should be simple enough. Second, the output should be simple, even only text. Third, it requires few keys to play and can be finished within few minutes or the game status can be saved. The role-playing games puzzle games, and strategy games satisfy the demand for these properties. Such kinds of games can be designed in the way of client-server model. When players start to play, the terminals display current game states first and ask for commands. Then players make choices. The game server receives commands and replies new states and new questions to players. The game is repeated by this way until game over.

The game procedure can be viewed as a finial state machine. Each command (request) transits the current game state (page) to enter next state. We can follow the steps below to develop WAP games.

1) Design the detail state transition diagram.
2) Implement the state transition diagram.
3) Layout the contexts according to each state.

Next, we need to deal with the session problem of WAP games. The session is an abstraction in order to provide logic continuity of the couple WAP request-response. There exist a set of techniques (Cookie, HTML Forms, URL rewriting, Java Servlet Session Tracking, and database) used to manage sessions [7].

Third, game servers need to access all the parameters instantly. Traditionally web applications are designed page by page, and thus parameters bypassing or accessed from database is very often. The database cannot provide faster speed than the memory does. Therefore, anyone who wants to build a WAP game with a lager scale should take in consideration of the following things:

- **An independent game server.** A server can run for a long time and keep the game world. All necessary data are stored in the memory, not in the database. Accessing local memory is much faster than querying the database. The server keeps the information

in its memory, and accesses the database if necessary.

- **Use WMLScript to reduce the computation load of servers.** Although the mobile phones have limited resources, using WMLScript to do simple calculation is still saving the communication time.

Last, to create a fair game, programmers must deal with the security problems. The stateless properties could introduce replay attack. The requests could be modified since the web pages could be saved on the storage of the clients. Ensuring the integrity of the communication is very important in game applications.

There are so many limitations on wireless environment. It is not an easy task to produce WAP games without the knowledge of WML, scripting language, database management, security, and algorithms of games. In order to ease this task, we design and implement a WAP Game Development Environment (WGDE) that has a WML page generator and a WAP game template. We call the sets of APIs "the WAP Game Foundation Class (WGFC)." Since the WAP games can be designed by finite state machine, we planned to develop a visual environment which lets programmers "drag and drop" the state transition diagrams. The platform will be described amply in the following sections.

3. The WAP Game Foundation Class

Since the late 1980s, the object-oriented approach to programming has been widely adopted. Programmers use software components done by others instead of developing all services by themselves. Reusing components in projects can greatly increase the speed of software development. Microsoft Foundation Class is such a product that has hundred classes, including applications, graphics, Internet services, database, containers, and so on. Window application developers use these classes according to what they need to finish projects in shorter periods.

We know that using a finite state machine is a more appropriate architecture for WAP games. If we want to transfer Web games to WAP games, the main job is to replace the HTML layout by WML layout. Although WML is similar to HTML, the conversion is not an easy task, especially in dynamic contents. The motivation of the Wireless Game Foundation Class (WGFC) is to provide a progression of classes that generate WML automatically, and to transparently deal with the problems discussed. This is first done in our past work [1].

3.1. The Architecture of the WGFC

The WGFC could be separated into two groups, including wireless game template classes, and tag-related classes. The WGFC architecture was shown in Figure 3.

- **Wireless game template classes.** This group contains five main classes, including WServlet, WLogin, WMenu, WWordFly, and WImageFly. All classes inherit HttpServlet class. The WServlet class provides some methods useful in wireless games, such as session, recently accessed page, WML tags objects containers, and so on. The WLogin class provides login templates. The WMenu class provides somekinds of menus templates. The WWordFly class provides flying messages templates. The WImageFly class provides flying images templates.
- **WML tags related classes.** In this group, all classes inherit the WTag class. All classes names are Wdeck, Wtimer, WImage, WText, WLink, WTextArea, WLinkList, WButton, WPButton, WAButton, WSButton, WForm, WInput, WSelect, etc. Packing WML tags into the form of JAVA classes helps game programmers develop wireless games whether they understand WML or not.

The WGFC focuses on the limitations we mentioned before and it has several advantages making programmers invest their games more.

- **Automatic Layout.** It is inefficient and is an impossible mission to test programs in all kinds of terminals when developing wireless games. This is because there are too many different screen attributes terminals. In order to make work done easier, the WGFC provides layout functions, just like JAVA layout manager. For example, FlowLayout methods display messages by the order that they are added into the container.
- **Divide contents automatically to fit the deck size.** The size of a deck is limited, such as 1.4K bytes in Nokia 7110. The limitation of content size also limits effects that could be used by game programmers. A programmer must take in consideration of the program logic and keep the size limitation in their mind. The WGFC has been added a feature that can produce contents smaller than a certain amount of bytes. When containers are fed with a lot of WML tags objects and beyond the limitation, the output content will be divided into several decks.
- **Support sessions.** The WGFC consists of an auxiliary database to store information about sessions and state of the application that can be recovered after unexpected disconnection. Game programmers call the function put() to store information into database and call the function get() to access what were stored.

- **Support security communication.** While the WGFC generates the WML pages, it encrypts the fields that will be sent back with the request. While receiving requests, the WGFC decrypts the fields and test their integrity. The encrypted data include serial number, checksum, and the original fields. Therefore, although users may see the source code, they cannot modify the fields.

We will discuss some design problems in detail in the following subsections.

3.2. The Stateless Problem

There exists a set of techniques such as Cookie, URL rewriting, session tracking, and HTML forms) design for bypassing the stateless nature of HTTP communication and for managing sessions. However, many terminals that support WAP still do not support Cookie technique until now, which is widely used in web applications. Meanwhile, URL rewriting could expose user's information and has cache problem, so it is not suitable for WAP games.

We solved this problem by using the session identifiers. Each WML page contains a session identifier session_id:

Current session_id = username + random_number

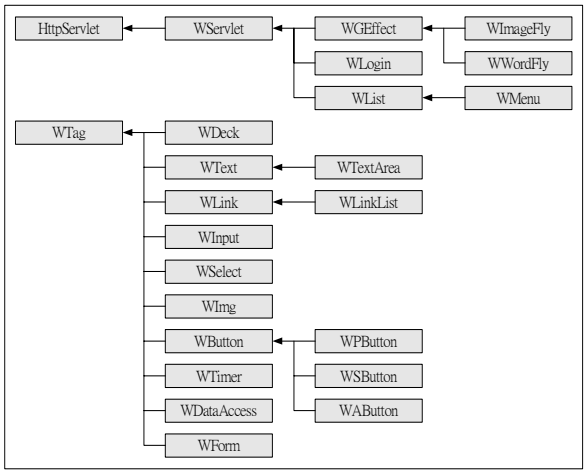

Figure 3. The WGFC architecture.

```
<anchor>
   <go href='C' method='post'>
      <postfield name='cmd' value='place;8;0'/>
      <postfield name='sid' value='wan1021146025279'/>
   </go>my command
</anchor>
```

Figure 4. A WML anchor with session identifier.

Figure 4 shows such an example. The session id is reassigned each time when player login. Any two players do not have the same usernames, and neither do the session identifiers. Each WML page generated by the WGFC contains session identifiers. Each time when players join the game, they get different session identifiers. The session identifiers not only identify players, but also ensure the requests does not come from a cached WML page before the game starts. Session lifetime is controlled by timeout mechanism. If one player does not send any request to the server in time, the session will be invalidated.

3.3. The Security Issues

It is not enough that just checking the session identifiers. There are still three problems we have to deal with. First, we must keep secret of the players' information. Second, we must keep the orders of the requests. Third, we must test the integrity of each request.

To solve the first problem, the WGFC sends data by "POST" method. The "GET" method implemented by URL rewriting could expose secret data, such as user's password. Besides, the "POST" requests are always sent to server without cache. It is defined in RFC2068.

For example, if the username and his password are passed by "GET"，the URL would be rewritten. If not specified, the WML would use "GET" by default. Figure 5 shows this situation. When the user submits the request to the server, the logged URL has the form of "http://localhost/check?id=wan&pw=mypwd." Clearly, the user's password is exposed in this case. Many Web servers log the requested URL, and Web browsers do, too. This is the reason why one should use "POST" method to send data.

```
<fieldset title="ID_PW">
  Enters id:<input name="id" title="id"/>
  Enters pw:<input type="password" name="pw" title="pw"/>
</fieldset>
<anchor title="Login">Login
   <go href="check">
      <postfield name="id" value="$(id)"/>
      <postfield name="pw" value="$(pw)"/>
   </go>
</anchor>
```

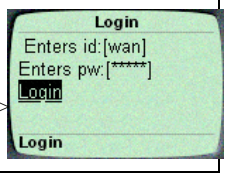

Figure 5. A WML page using "GET" method.

Now we will show how to deal with the next two problems. The requests could be reordered or replayed because of the WAP stateless nature. The worse situation is that the request is modified. To solve these problems, each session has its serial number. At beginning, the serial number is the random number appended to the username. Each time when a page is sent to the client, all its command string will be encrypted by this serial number. The server also stores the commands in the player's session for further checking when they back. Commands will be decrypted and compared to the stored ones. The serial number is changed after the successful test. The original serial number is treated

as a random seed to generate next serial number.

This method ensures that the requests and responses in the right order and keeps its integrity. We encrypted the command by calculate the exclusive-or value of the command and serial number. Figure 6 shows the encrypted version of Figure 4. The original value of "cmd" is "place;8;0." After using the DES algorithm to encrypt the field, we get "3bb7a1656b930b7e." (hexadecimal representation)

```
<anchor>
  <go href='C' method='post'>
    <postfield name='cmd' value='3bb7a1656b930b7e'/>
    <postfield name='sid' value='wan1021146025279'/>
  </go>my command
</anchor>
```

Figure 6. A WML anchor with encrypted command.

4. Conclusions and Future Work

We have provided a platform for the WAP game development. This platform can reduce the time needed to overcome the problems that all developers must face to. First, programmers can develop WAP games easily by using the WGFC without much knowledge about the syntax of WML. Second, the WAP game template deals with the common issues like deck size, session and security, so programmers can concentrate on their games more.

The WAP games can be modeled as finite state machines. Based on the theory of visual programming, we are trying to implement a visual environment, like Macromedia Authorware, for WAP game development. Our goal is to make the procedure of the WAP game development as follows:

1) Design the detail game state transition diagram and draw it in this environment.
2) Use visual tools to layout contexts.
3) Modify the automatically generated code.

Due to the stateless nature of WAP and HTTP, currently we use database to store game states. However, this method suffers from inefficiency. And thus it is not suitable to build a larger game such as MUD on WAP. Now we are studying the techniques needed in large scale online games. A game server can maintain complete game status and all online players' information without having to access the database frequently, and thus it will improve the performance.

References

[1] Chu-Shiang Shu, "WGFC – A WAP Game Delopment Platform ," Department of Computer Science and Information Engineering, National Chiao Tung University, Thesis of Master, advised by Wen-Nung Tsai, June 2001.

[2] A. Fasbender and F. Reichert, "Any Network, Any Terminal, Anywhere," *IEEE Personal Communications*, pp22-30, 1999.

[3] A. Schmidt and A. Takaluoma, "Context-Aware Telephony over WAP," *Personal Technologies*, pp.225-229, 2000.

[4] E. Kaasinen, M. Aaltonen and T. Laakko, "Defining User Requirements for WAP Services," *Human-Computer Interaction*, pp.33-37, 1999.

[5] E. Kaasinen and M. Aaltonen, "Two Approaches to Bringing Internet Services to WAP Devices," *Computer Network*, pp.231-24, 2000

[6] T. Lilja, "Mobile Energy Supervision," *Telecommunications Energy Conference, 2000. INTELEC. Twenty-second International*, pp.707-712

[7] M. Cannatard amd D. Pascum, "An Object-based Architecture for WAP-compliant Applications," *Database and Expert Systems Applications, 2000. Proceedings.* 11th International Workshop on, pp.178-185, 2000.

[8] M. Metter and R. Colomb, "WAP Enabling Existing HTML Applications," *User Interface Conference, 2000. First Australasian,* pp.49 –57, 1999.

[9] K. M. Goeschka and M. W. Schranz, "Client and Legacy Integration in Object-Oriented Web Engineering," *IEEE Multimedia*, pp.32-41, Jan.-March 2001.

[10] A. K. Ghosh and T. M. Swaminatha, "Software Security and Privacy Risks in Mobile E-Commerce," *Communications of the ACM*, Volume 44, Issue 2. pp. 51-57, February 2001.

[11] g M. Jones and G. Marsden, "Improving WEB Interaction on Small Displays," *Computer Network*, ppt.1130-1137, 1999.

[12] BEA Htmlkona, http://e-docs.bea.com/wls/docs61/htmlkona/

[13] Sun Microsystems, URL:http://java.sun.com/

[14] WAP Forum, URL:http://www.wapforum.org/