International Journal on Artificial Intelligence Tools
Vol. 18, No. 1 (2009) 141–161
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

SOA-BASED CONTENT DELIVERY MODEL FOR MOBILE INTERNET NAVIGATION

STEPHEN J. H. YANG

Department of Computer Science and Information Engineering
No. 300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan (R.O.C.)
jhyang@csie.ncu.edu.tw

JIA ZHANG

Department of Computer Science, Northern Illinois University, DeKalb, IL, USA
jiazhang@cs.niu.edu

JEFFREY J. P. TSAI

Department of Computer Science, University of Illinois, Chicago, IL, USA
tsai@cs.uic.edu

ANGUS F. M. HUANG

Department of Computer Science and Information Engineering
No. 300, Jhongda Rd., Jhongli City, Taoyuan County 32001, Taiwan (R.O.C.)
fmhuang@csie.ncu.edu.tw

Received 13 August 2008
Revised 8 October 2008

This paper presents a Service Oriented Architecture (SOA)-based content delivery model to facilitate mobile content delivery. The main contribution of this paper is the design and development of an SOA-equipped content delivery system based on a context-driven, access-controlled, profile-favored, and history-maintained (CAPH) model. We embody the generic model-view-controller (MVC) model to support a dynamic content adaptation technique based on mobile users' contextual environments. Self-adaptable presentation objects and modules are modeled as universal Web services resources, so that their interactions are formalized into Web services operations for high interoperability. Experimental results demonstrate that our proposed SOA-based model makes it easy to configure and construct a flexible Web content delivery system on the mobile Internet.

Keywords: Mobile Internet; web services; content adaptation; handheld devices.

1. Introduction

Although bringing unprecedented flexibility and mobility, mobile content delivery poses big challenges to Web content delivery. Most of the existing Web content is not originally designed for being displayed on handheld devices. Instead, their default settings and page layouts are mostly designed for presentation on desktop computers.

Therefore, direct content delivery without layout adjustment and content adaptation often leads to information disorganized on handheld screens, let alone it requires that users constantly move scroll bars before perceiving a complete piece of information.

Figure 1 illustrates differences of applying Web content adaptation with a comparison of Web browsing between conventional and handheld devices. The middle of Figure 1 shows the yahoo home page (http://www.yahoo.com/) on a desktop browser. The left-hand side of Figure 1 shows how this page is shown on a PDA screen without content adaptation (upper-side) and with content adaptation (lower-side). The right-hand side of Figure 1 shows how the same page is shown on a wireless phone screen without content adaptation (upper-side) and with content adaptation (lower-side). As shown in Figure 1, without content adaptation, users of PDAs and wireless phones have to move scroll bars left and right, up and down in order to view the whole Web page. In contrast, our content adaptation obviously provides better content presentation to the users by transforming the Web page into a column-wise presentation, so that users only need to move scroll bars in one direction (up and down) instead of in two directions.

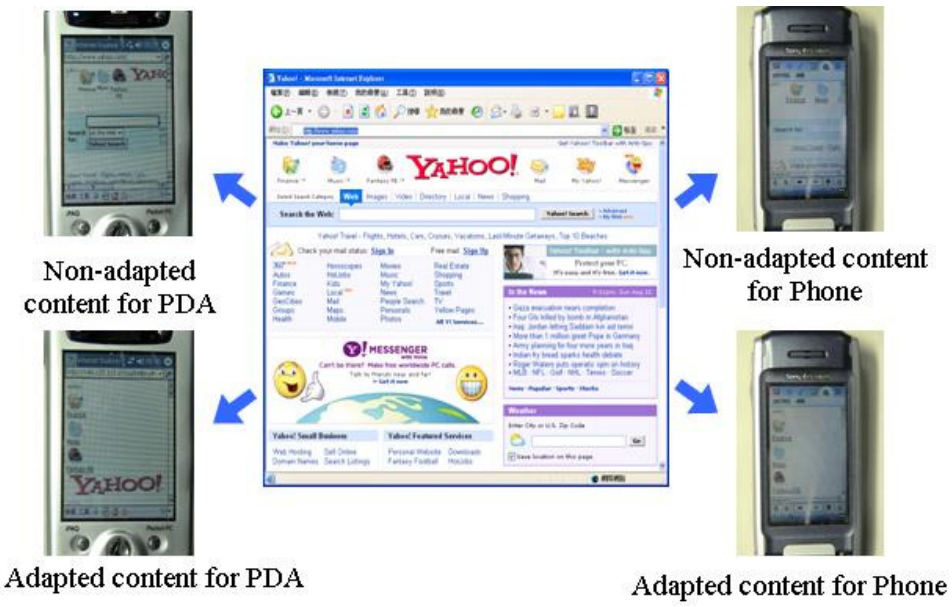

Fig. 1. Snapshots of adapted Web page for NB, PDA, and Phone.

As a result, a technique is needed to deliver adaptive content in a proper format to any device through any network at anytime from anywhere. Considering that Web content typically undergoes frequent changes, the conventional approach of preparing different versions (formats) of the same content for mobile devices is neither practical nor feasible for Web content delivery. To deliver personalized and adaptive content according to users' situated environment, an underlying model is critical to conduct proper content

adaptation after considering users' contextual information.[1] In this paper, a user's computing context (or context in short)[2-4] refers to any environmental property (including devices, network, location, and time) that may affect the user's mobile access of Web content.[5] Meanwhile, considering the limited bandwidth and resources available in a mobile environment, the computation overhead due to content adaptation has to be taken into consideration.

In our previous research, we created a mechanism that takes HTML documents as input, automatically identifies semantically coherent presentation groups, and generates adapted content based on rules and receiving contexts.[6-9] This paper reports how we apply Service-Oriented Architecture (SOA) and Web services technology to design and develop a content adaptation system to facilitate mobile content delivery. Service Oriented Architecture (SOA) is a newly emerged architectural model in recent years, which aims to guide in building software applications with higher reusability, flexibility, extensibility, and interoperability.[10] Applying the SOA concepts and techniques, we model self-adaptable presentation objects as universal Web services resources, so that their interactions are formalized into Web services operations for high interoperability. The way we build our system can be used to guide in constructing content adaptation tools and services, which can be either plugged into existing content delivery platforms or act as a universally accessible Web service.

In this paper, we present a context-driven, access controlled, profile-favored, and history-maintained MVC model (CAPH-MVC) as our backbone to support a dynamic content adaptation technique based on users' contextual environments. Our model aims to improve universal Web content accessibility and reduce content access time, while maintaining semantic coherence of the original content. Meanwhile, we adopt the concept of Service Oriented Architecture (SOA) to enhance the reusability and configurability of the proposed CAPH-MVC for more effective and efficient Web content delivery over the mobile Internet.

The remainder of the paper is organized as follows. We will first discuss related work. Then we will present our CAPH-MVC model and discuss the dynamic content adaptation manager and its supporting CAPH managers. Afterwards, we will discuss modeling and implementation details, and present conducted experiments and discussions. Finally, we will make conclusions.

2. Related Work

The conventional framework of providing Web content supporting mobile devices is to prepare different versions (formats) of the same content for various mobile devices. For example, a Web page holds one HTML version supporting desktop devices and one Wireless Markup Language (WML) version supporting wireless devices. This approach is straightforward but labor-intensive and inflexible. Content providers have to prepare different layouts and formats for the same Web content, which results in tremendous overhead. Even worse, any change in the content may result in consequent changes in every related version, which is highly inflexible and may easily cause inconsistency.

Considering Web content typically undergoes frequent changes, this traditional approach is neither practical nor feasible for content delivery.

Thus, there appear various content publishing tools that provide automatic content adaptation[11] facilities that transform Web pages into proper formats before delivering them to different receiving devices. Typical tools include Oracle application server wireless,[12] Sun Java system portal server mobile access,[13] and WebSphere transcoding publisher.[14] However, these publishing tools require that the adaptable content be developed using the same tools and platforms from the beginning. Requiring that all Web content be developed in a formalized way, maybe during a rather long period of time, is neither feasible nor even desirable.

Therefore, in recent years many researchers have been building content adaptation prototypes that transform existing Web content into various formats. Among them, Phan, Zorpas, & Bagrodia[15] propose a middleware, called Content Adaptation Pipeline (CAP), to perform content adaptation on any complex data types, in addition to text and graphic images. They use eXtensible Markup Language (XML) to describe all elements in a content hierarchy. However, their work is based on an assumption that the XML data model is obtained ahead of time. Berhe, Brunie, & Pierson[16] present a service-based content adaptation framework. An adaptation operator is introduced as an abstraction of various transformation operations such as compression, decompression, scaling, and conversion. Their work shows a proof-of-concept of Web content adaptation; however, the actual implementation is still in a primary phase. How to map from constraints to adaptation operators remains unsolved. Lee, Chandranmenon, & Miller[17] develop a middleware-based content adaptation server providing transcoding utilities named GAMMAR. A table-driven architecture is adopted to manage transcoding services located across a cluster of network computers. Their approach allows incorporation of new third-party transcoding utilities. Lemlouma and Layaida[18] propose an adaptation framework, which defines an adaptation strategy as a set of description models, communication protocols, and negotiation and adaptation methods.

In our previous research, we studied five content adaptation techniques[7,8]: resizing (a technique to reduce content's shape to fit within the smaller screen size of portable devices), column-wise (a technique to transform content's presentation layout from multiple-columns to a single-column), thumbnail (a technique to replace a large area of image with small icons to fit within a smaller screen size), replacing (a technique to replace rich media with text or voice), and transcoding (a technique to transform media types with different modalities and fidelities). Their issue is that content adaptation may mess up the organization of a content page and lead to misunderstanding. Without retaining semantic coherence and relationships between semantic units, In Ref. 8, we present a unit of information (UOI)-based content adaptation method, which automatically detects semantic relationships among comprising components in Web contents, and then reorganizes page layout to fit handheld devices based on identified UOIs. In Ref. 9, we present an ontology-based context model supported by context query and phased acquisition techniques. In Ref. 6, we present a JESS-enabled context

elicitation system featuring an ontology-based context model to formally describe and acquire contextual information pertaining to service requesters and Web services.

In short, our previous works create a mechanism that takes HTML documents as input, automatically identify semantically coherent presentation groups, and generates adapted content based on customizable rules and receiving contexts. In contrast, this paper presents how we apply the SOA concepts and Web services technology to design and develop our system that can be used to guide in constructing flexible and extensible content adaptation tools and services.

3. Content Delivery Model

In this section, we will present our content delivery model as the backbone of our system. Since we aim to support appropriate Web content delivery and presentation onto various mobile devices, in addition to traditional desktops and laptops, we need an underlying infrastructure to enable this feature of automatic adaptability. Our solution is a modularized, SOA-oriented content delivery model as a foundation for fulfilling dynamic Web content delivery on-demand.

3.1. *CAPH-MVC model*

The most well known architectural model for building adaptive content delivery is the Model-View-Controller (MVC) model.[19] It separates code for data access, business logic, and data presentation and user interaction into three loosely coupled layers of components. As we intend to support different views on various receiving devices for the same Web content, it is natural for us to apply the MVC model in our design. As shown in Figure 2, specific Web content act as the model; different receiving devices (desktops, PDAs, and mobile phones) reflect different views of the common Web content; a controller resides in between to manage the transformation of the Web content into different views under certain circumstances.

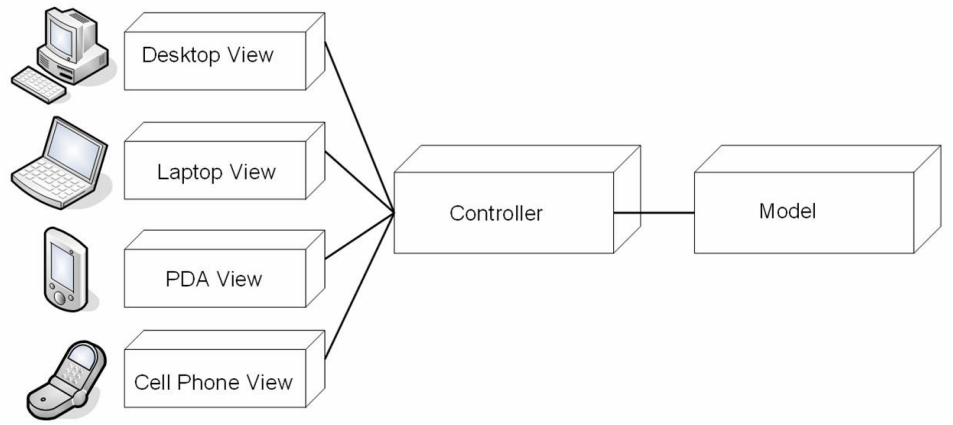

Fig. 2. Direct application of MVC model.

In general, the traditional MVC model can be represented as a controller function as follows:

$$V = F(M) \mid T_R$$

where V: denotes the view of a certain type of content (M) at a moment (T_R).
F: denotes the controller function.
M: denotes a certain type of content.
$F(M)$: denotes the result of the controller function applied to the content (M).
T_R: denotes the context of the receiver type at a moment.

However, the basic MVC model is too generic and coarsely grained; therefore, it can only provide very high-level guidance for developers to design a content delivery solution. Specific to mobile content delivery, we embody the controller in the traditional MVC model into a context-driven, access controlled, profile-favored, and history-maintained MVC model (CAPH-MVC). The four sub-controllers are summarized from our previous work on mobile content delivery.[6-9] Figure 3 illustrates its concept.

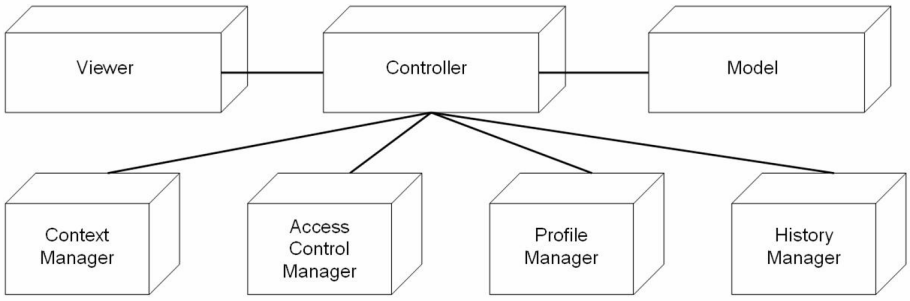

Fig. 3. CAPH-MVC model.

The controller function of our CAPH-MVC model can be represented as follows:

$$V = F(M) \mid (T_R \wedge C_R \wedge A_R \wedge P_R \wedge H_R)$$

where V: denotes the view of a certain type of content (M) at a moment (T_R).
F: denotes the controller function.
M: denotes a certain type of content.
$F(M)$: denotes the result of the controller function applied to the content ($F(M)$),
T_R: denotes the context of the receiver type at a moment.
C_R : denotes receiver context.
A_R : denotes receiver access control.
P_R: denotes receiver profile.
H_R: denotes receiver history.

3.2. *Presentation module*

Based on our extended CAPH-MVC model, we establish a presentation module to embody our design as shown in Figure 4. Content view represents various presentations on different receiving devices; content represents the original Web content designed for desktop presentations. These two parts act as the viewer and the model defined in the original MVC model (in Figure 3), respectively. The content adaptation manager extends its counterpart controller, by exploiting support from four components: context manager, access control manager, profile manager, and history manager. These components support the content adaptation manager for adaptive content delivery by providing proper contextual information, access privileges, user profiles, and historical information. In short, our CAPH-MVC model offers a presentation pattern supporting typical mobile Internet content delivery.

The content controller is supported by four components: context manager, access control manager, profile manager, and history manager. At the moment of access, the context manager gathers environmental features of the receiver including device type, network condition, moving situation, and so on. The access control manager provides a fine-grained access control for various parts of the content based on receiver identity. The profile manager uses receiver profile to provide personalized content delivery. The history manager uses the receiver's historical access data to help deliver proper content with appropriate format.

As shown in Figure 4, a content rule manager is separated from the content controller to enable configurable and extensible content management under the control of various rules. To further enhance performance of content adaptation, a cache component is introduced. In addition, an independent content adaptation manager is established, which is dedicated to converting content into appropriate formats for proper presentation onto various receiving devices.

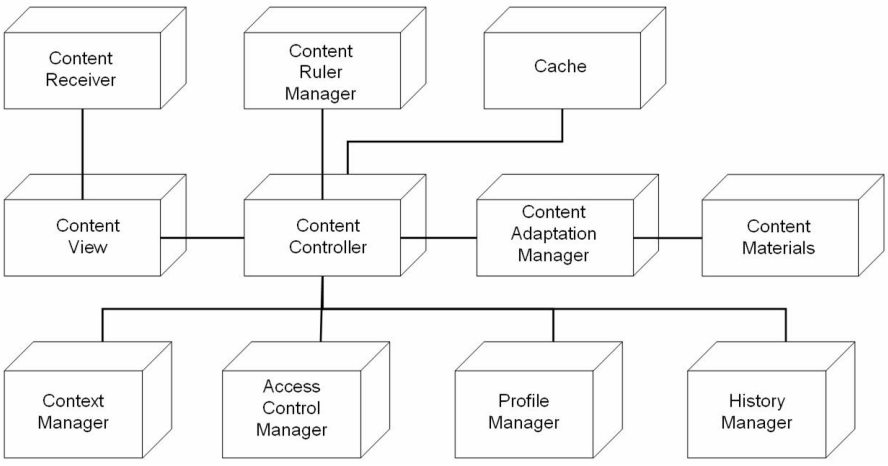

Fig. 4. Presentation module based on CAPH-MVC model.

As shown in Figure 4, a content receiver component is introduced into the presentation module, representing an external receiver, either a program or an individual who requests a service from the presentation module. The reason why we define it as an integral part of the presentation module is to enhance configurability. A content receiver represents a class of receivers, which is associated with a 4-tuple: (receiver type, receiver profile, receiver context, receiver identity). Receiver type specifies the type of the receiving device (e.g., PDA, mobile phone, or desktop); receiver profile specifies specific attributes of the receiver; receiver context specifies specific environmental features of the receiver; receiver identity specifies the role and privilege of the receiver. Each individual content requester is an instance of a content receiver class with proprietary states of the above four properties.

3.3. *CAPH managers*

As shown in Figure 4, the four CAPH managers cooperate to gather and elicit runtime information to help the content adaptation controller make appropriate decisions. Context manager helps gather contextual information; access control manager helps gather security-related information; profile manager helps gather preference information; history manager helps gather historical data.

To facilitate generic context-aware services delivery, we proposed an ontology-based context model to formally define context description pertaining to both service requesters and services.[8] Two types of context ontology are developed for describing the circumstances of requesters and services, respectively: requester ontology and service ontology. Centered on a list of profiles, each ontology defines a hierarchical system of Web Ontology Language (OWL)-based generic construction units.

Applying our context model to Web content delivery, users using various receiving devices can be viewed as service requestors; Web content delivery services can be viewed as services. Here we extend our two previous generic ontologies in Ref. 9 shown as follows to cover application-specific contextual requirements. Note that we use a pseudo-formal notation to make the definitions self explain thus easy for readers to understand.

Service_ontology =

{Profile, QoWS, Environment, Devices}

Service Profile = {name, id, description, input, output, pre-condition, effect}

QoWS = {Functional requirement, non-functional requirement}

Environment = {Network channel, Situation}

Network channel = {wired, wireless}

Situation = {normal, meeting, walking, driving}

Devices = {hardware, software}

Requester_ontology =

{Profiles, Preferences, QoWS, Environment, Devices}

Profiles = {Personnel, Location, Calendar, Social}

Personnel_profile = {name, role, id, email, accessibility}

Location_profile = {office, building, home}

Calendar_profile = {owner, event , time, attendee^{+}, location}

Social_profile = {owner, partner*}

QoWS = {Functional requirements, non-functional requirements}

Environment = {Network_channel, Situation}

Network_channel = {wired, wireless}

Situation = {normal, meeting, walking, driving}

Devices = {hardware, software}

We provide ontologies for both service providers and service requesters. The former defines suitable contextual requirements to deliver the service; the latter defines user contexts. Particularly, we added access control-related profiles into both the requester ontology and service ontology. The necessity is obvious: only registered users have access to corresponding Web content. The following segment shows their simplified profile specifications. Extended service ontology contains an access control profile comprising a set of attributes such as service id, year, and requirements. Extended service requester ontology contains an access control profile comprising a set of attributes such as requester id, requester name, and social security number.

//In service ontology:
Access_control_profile = {service_id, year, requirements}
//In service requester ontology:
Access_control_profile = {requester_id, requester_name, social_security_number}

To acquire contextual information at run time, we may adopt a three-step procedure as we discussed in Ref. 9: a form-filling phase constructs service requester ontology based on requester inputs; a context detection phase collects and analyzes requesters' dynamic contextual information; a context extraction phase derives contextual information from requesters' preferences and profiles.

We bind access control facility at the level of structure layer of our OSM model. If a Web content receiver has access to a medium object, he/she may receive its version in appropriate modality and fidelity based on the contextual situation at the moment. The reason why we do not limit access control to an entire Web content is to enhance content

reusability. Therefore, each medium object is associated with a list of services to which it belongs at the moment. This list can help the medium object decide whether a service requester has access to it or not.

Note that such an association relationship between a presentation object and a Web site has its lifecycle.[20] For example, consider a video clip belongs to a service provider. A user who put subscription to the service provider has access to this video clip. After the subscription ends, however, the user should lose the access to the video clip. Internally, this video clip should be removed from the access list associated with the video clip. An event-driven pattern is adopted by our access control manager. In case an event happens (e.g., a time clock is clicked to enter subscription period), an access control action is triggered (e.g., a video clip information is associated with corresponding presentation objects). We also specify and realize a role-based access control mechanism, where service provider roles could edit contents while service requester roles could not. It is beyond the scope of this paper so that it will not be discussed here.

The profile manager handles various services and service requester preferences and profiles. For service requesters, we focus on four types of profiles: personnel, location, calendar, and device profile. A personnel profile describes a service requester's identity and preference; a location profile describes a service requester's physical location; a calendar profile describes what, when and where an activity occurs as well as who are with the requester; device profile describes the receiving device of the service requester. For services, we focus on the attributes of Web content such as its name, id, description, prerequisite, and effect. The profile manager can be viewed as a critical supporting service to the context manager.

The history manager records and keeps the history of service requests associated with service requesters who registers and uses the system. Based on the historical information, we can analyze the requesting behaviours and requesting patterns that are important references for building service requesters' preferences.

4. Content Adaptation Manager

In our CAPH-MVC model, the content adaptation manager is a key component, since many other components exist to help the content adaptation manager decide appropriate transformation requirements and actions. Therefore, in this section, we will present our design details of the content adaptation manager, its internal structure and design strategy.

4.1. *Content adaptation infrastructure*

We identified two major research challenges of content adaptation: (1) how to detect and maintain semantic coherence relationships among composing elements of the content in the process of transformation; and (2) how to enable dynamic content adaptation according to dynamic user requirements and environmental features. Our strategy is to model the content adaptation manager as a self-contained, encapsulated services system,

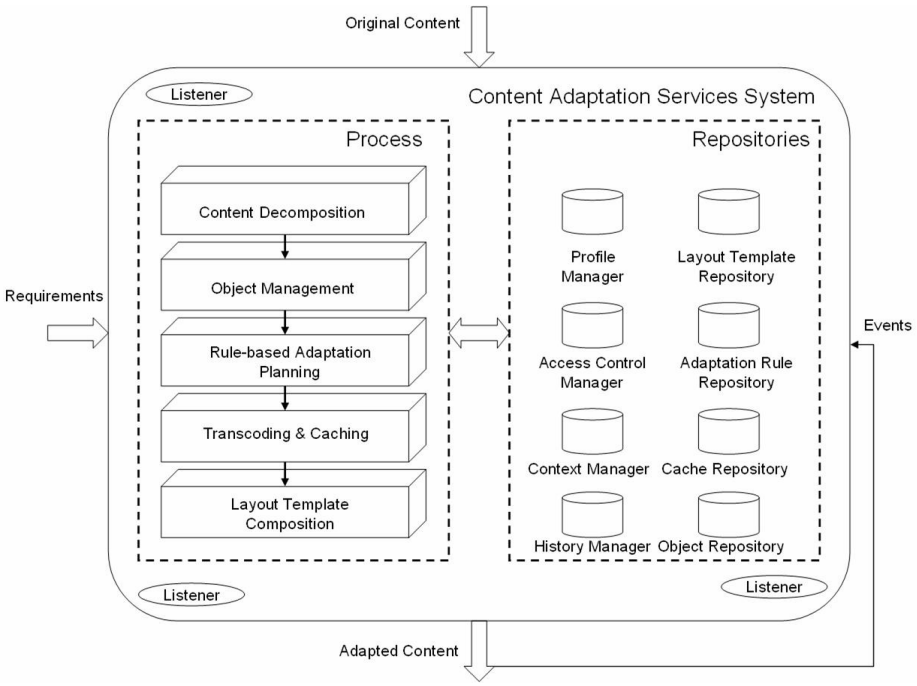

Fig. 5. Content adaptation manager.

featured as a feedback system, as shown in Figure 5. Such a system contains internal states and reacts to surrounding environmental changes triggered as events.

We define a content adaptation manager *CAM* as a 6-tuple:

$$CAM = \langle Inputs, Outputs, Requirements, Process, Repositories, Listeners \rangle,$$

where *Inputs*: denote original content in its original format.
Output: denote the adapted content in its proper format.
Requirements: denote various required features, e.g., receiving device, network bandwidth, receiver status, and so on.
Repositories: denote information storages managed by supporting CAPH managers.
Listeners: denote event listeners that monitor and detect changes from surrounding environments.
Process: denotes a stepwise content transformation procedure (workflow), featured with the ability of semantic coherence retaining.

As shown in Figure 5, repositories comprise four databases (layout template repository, adaptation rule repository, cache repository, and object repositories) controlled by their corresponding CAPH managers (context manager, access control manager, profile manager, and history manager). With the assistance of Listeners, our

content adaptation manager is able to dynamically adjust transformation strategies according to runtime requirements and environments. The content adaptation process represents a workflow comprising five steps: content decomposition, object management, rule-based adaptation planning, transcoding and caching, and layout template composition. The content adaptation process will be discussed in the following section.

4.2. *Adaptation planning, transcoding, and caching*

According to the information gathered by the context manager, access control manager, profile manager, and history manager, an adaptation planner retrieves the object tree to select appropriate modality objects and fidelity objects for corresponding structure objects. This type of content adaptation is called static adaptation. In case no applicable modality or fidelity objects can be found in *object repository* for a medium object, a *dynamic transcoding* process is invoked. This type of content adaptation is called dynamic adaptation. The transformed objects are cached and stored into the *object repository* for future reuse.

Each type of medium transcoding is associated with a tool pool, where any third-party transcoding services can be easily plugged in through Web services-based registration. In our implementation, we have exercised and adopted a variety of third-party transcoding tools to transform objects into various modality and the fidelity, including VCDGear[21] for video transformation, PictView[22] for image conversion, and JafSoft[23] and Microsoft Reader[24] for text-to-speech (or vice versa) conversion. These tools have been linked into our system as default transcoding tools.

Dynamic content adaptation is unavoidably accompanied with additional transcoding overhead at run time. From a performance perspective, our previous work[8] yielded a caching mechanism similar to that is introduced by Kinno, Yukitomo, & Nakayama.[25] In contrast to their approach, we only cache the transcoded objects through our *cache management*, instead of caching the entire adapted content. One challenge is how to ensure proper management of object trees, when transcoded objects need to be inserted and removed from the object trees. Our solution is to treat every node in an object tree as a universal WSRF resource, which encapsulates interfaces for dynamic related node insertion and removal. Therefore, management of such an object tree is distributed to all encompassing nodes. It should be noted that our cache management not only reduces transcoding overhead, but also significantly saves transmission bandwidth. This is because the sizes of the transcoded objects are typically reduced; therefore, receiving devices such as PDAs and mobile phones may largely benefit from reduced amount of transmission data.

Runtime adaptation planning and transcoding rely on an underlying rule base, which is dynamically constructed and maintained. It should be noted that rules here are designed in a way that they can be configured and reconfigured. The verification of its soundness and completeness is triggered by events and is based on our previously developed truth maintenance mechanism.[26] Detailed JESS-based adaptation planner design and implementation can refer to our previous paper.[6]

5. System Modeling

In this section, we present how we apply the Web services technology to model and realize our CAPH-MVC model, toward ensuring reusability, interoperability, flexibility, extensibility, and adaptability.

5.1. *Presentation object modeling*

Web content typically comprises a set of presentation objects, or simply objects. We model a specific piece of Web content as a 4-tuple:

$$Content = \langle Obj, Str, Mod, Fid \rangle,$$

where *Obj*: denote a medium object encompassed in a Web page.
Str: denote the object's structure. $Str \in \{object\_cluster, unit\_of\_information\}$
Mod: denote the object's modality. $Mod \in \{video, audio, text, \ldots\}$
Fid: denote the object's fidelity. $Fid \in \{jpeg, bmp, mp3, wmv, midi, \ldots\}$

According to our previous research,[7] Web page content is organized into a three-layer structure, namely structure layer, modality layer, and fidelity layer. The structural layer comprises the objects contained in the content; the modality layer comprises possible presentation types for each object; the fidelity layer further specifies possible presentation formats for each presentation type. In short, this model helps organize objects with possible presentation versions of a given content page.

As highlighted in Figure 6, each node (in each layer) in such a hierarchical tree-like structure is stored as a universal resource using Web Services Resource Framework (WSRF)[27] technique. (Applying WSRF in CAPH-MVC modeling will be discussed in

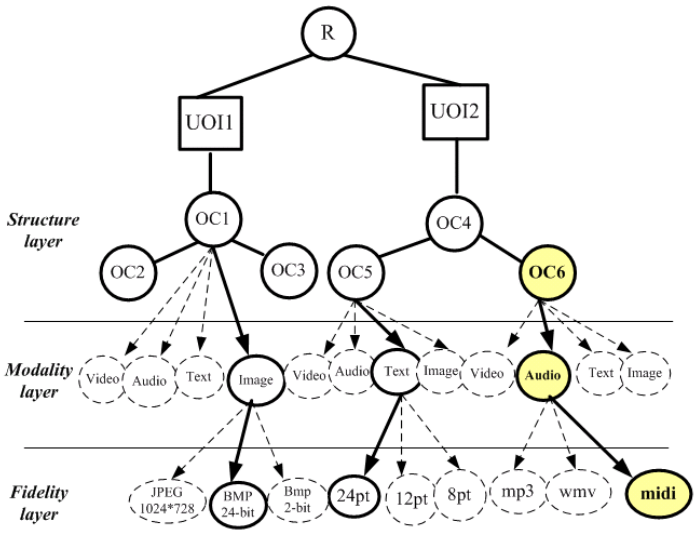

Fig. 6. Presentation object model.

detail in the next section.) Dynamic node creation and association is realized through Web services interfaces.

5.2. *Modeling presentation module*

Services computing has been widely admitted as the next generation of Web technology.[10] It provides a uniform and loosely coupled integration framework to increase cross-language and cross-platform interoperability for distributed computing and resource sharing over the Internet. Its central concept of Service Oriented Architecture (SOA) represents a novel way of decoupling the concerns between business functions and actual implementations.[28] Nowadays the Web services paradigm is considered the best enabling technology of SOA and Services Computing. In our research, to enhance reusability and configurability, we apply SOA and Web services technology to model the components of the presentation module.

We had two options to model the components, either in a stateless mode or in a stateful mode. The former mode can be modeled in the ad hoc industry standard Web Services Description Language (WSDL) and does not capture and maintain states of a service. This is probably enough for a read-only component, such as the content transformation manager. However, to provide personalized and history-maintained content delivery services, the stateful Web services mode is a better option. To be consistent, we decided to model all components in our presentation module in a stateful manner.

Initiated by IBM, Computer Associates (CA), Oracle, and other collaborators, WSRF[27] defines a system of specifications for managing and accessing stateful resources using Web services. Being an XML-based presentation method to capture resources, WSRF contains four sets of specifications: WS-ResourceProperties, WS-ResourceLifetime, WS-BaseFaults, and WS-ServiceGroup. These sets of specifications enable access to internal states of a resource via Web service interfaces, i.e., data values that persist across and evolve as a result of Web services interactions. In addition, WSRF supports dynamic creation of resource properties and associated values. In other words, WSRF describes how the state of a WS-Resource is made accessible through a Web service interface, and defines related mechanisms concerned with WS-Resource grouping and addressing.

Figure 7 shows a segment of a most simplified example resource properties document "ContentRuleManager". In order for a service requestor to know that the "ContentRuleManager" defines the WS-Resource properties document associated with the Web service, the WS-Resource properties document declaration is associated with the WSDL portType definition in the WSDL definition of the Web service interface, through the use of a standard attribute resourceProperties. As a result, as shown in Figure 7, the portType, with the associated resource properties document, defines the type of the WS-Resource.

It should be noted that a stateful service requires more coding and additional processing resources. Therefore, it typically has a definite impact on the performance of

the service (for example, cost, configurability and re-configurability of the service). Furthermore, it takes longer time to develop the mechanism and the application. Moreover, it may affect the scalability of the service. We will study the related overhead in later sections.

```
<!-- Association of resource properties document to a portType -->
<wsdl:definitions
   xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-services/ws-
resourceProperties"
   xmlns:tns="http://sc.org/ContentRuleManager">
…
   <wsdl:types>
      <xs:schema>
         <xs:import namespace="http://sc.org/ContentRuleManager"
               schemaLocation="…"/>
      </xs:schema>
   </wsdl:types>
…
   <wsdl:portType name="GetRule"
      wsrp:resourceProperties="tns:ContentRuleManager">
      <operation name="getPrice"/>
…
   </wsdl:portType>
</wsdl:definitions>
```

Fig. 7. A segment of an example stateful resource definition using WSRF.

5.3. *Interaction patterns in BPEL*

Each component in our presentation module is modeled as an atomic stateful Web service. The interactions between the components thus can be conducted through standard Simple Object Access Protocol (SOAP). Under certain circumstances, these components can be easily grouped together into a composite Web service dynamically. For example, upon request, all components in Figure 4 may collaborate to generate an appropriate view for the corresponding content. A workflow, or interaction pattern, is adopted to coordinate the interactions between the components. For instance, upon request, the content controller checks the content rule manager. Then it may query the context manager, access control manager, profile manager, and history manager before forwarding the request to the content adaptation manager.

We adopt ad hoc industry standard Business Process Execution Language for Web Services (BPEL4WS)[29] to define the interaction workflows between components. Using standard BPEL to describe the interactions, we allow flexibility and extensibility, as users can change the workflow definitions thus change content adaptation handling.[30] To show how we use BPEL4WS to describe the interactions among the components, we use a very

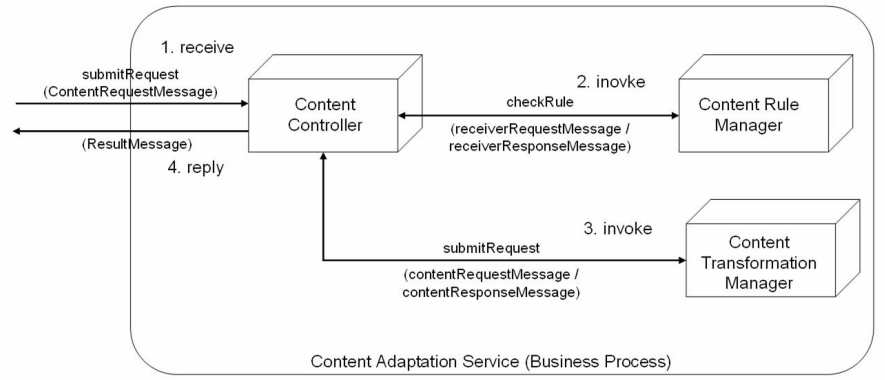

Fig. 8. Simplified content controller business process.

```
<process name="contentDeliveryService"…>

<!—Define process -->
  <sequence>
    <receive partnerLink="contentRequest"
        portType="lns:contentRequestPT"
        operation="submitContentRequest"
        variable="ContentRequest">
    </receive>

    <invoke partnerLink="rule"
        portType="lns:rulePT"
        operation="submitRule"
        inputVariable="RuleRequest"
        outputVariable="RuleResponse">
    </invoke>

    <invoke partnerLink="transformation"
        portType="lns:transformationPT"
        operation="submitTransformation"
        inputVariable="TransformationRequest"
        outputVariable="TransformationResponse">
    </invoke>

    <reply partnerLink="contentDelivery"
        portType="lns:contentRequestPT"
        operation="submitContentRequest"
        variable="Result"/>
  </sequence>
</process>
```

Fig. 9. A segment of BPEL definition of interaction patterns between components.

simplified example as shown in Figure 8. Receiving a request, the content controller consults with the content rule manager, and then invokes the content adaptation manager. After receiving the results, the content controller returns the results to users.

As shown in Figure 9, after the *receive* activity, the process invokes two Web services sequentially, each being delimited using an *invoke* tag. First, the process invokes the operation *submitRule* from the portType *rulePT*, with an input message *RuleRequest* (i.e., *RuleRequestMessage*) and an output message *RuleResponse* (i.e., *RuleResponse*Message). Then the process invokes the operation *submitTransformation* from the portType *transformationPT*, with an input message *TransformationRequest* (i.e., *TransformationRequestMessage*) and an output message *TransformationResponse* (i.e., *TransformationResponse*Message).

The fourth and the last activity is a *reply* activity, which allows the business process to send a message in reply to the requestor. Once a reply activity is reached, the four-element tuple <partnerLink, portType, operation, variable> is used to send the result back. In Figure 9, the *reply* activity invokes the *getResult* operation from the *contentRequestPT* portType with the variable *Result* (i.e., *ResultMessage*). Note that the combination of a pair of *receive* and *reply* forms a request-response operation on the WSDL portType for the process, in this example *contentRequest* operation in the portType *contentRequestPT*.

6. Implementations And Experiments

6.1. *Implementations*

We have implemented a CAPH content delivery prototype (VCAProxy) as a proof-of-concept. VCAProxy was developed on a server machine with an Intel Xeon 3.0GHz CPU and 3.5GB memory. The operating system is Microsoft Windows XP installed with software platforms including: proxy server, JAVA JVM and MySQL as a database management system. The client devices include two types: HP iPAQ Pocket PCs h5500 with Microsoft Windows C.E. 3.0 operating system and Sony Eriksson P900 smart phones with Symbian operating system. The browser is Microsoft Internet Explorer.

We chose Java as our programming language to develop our system, mainly due to its cross-platform compatibility. The architecture of our system is based on proxy. After examining several open-source Java proxy candidates, we selected Muffin[31] as our proxy server. Muffin is a relatively light-weight proxy supporting PHP and AJAX. It can be executed on Unix, Windows 95/NT, and Macintosh. It also supports several network protocols such as HTTP/1.0, HTTP/1.1, and SSL. We built VCAProxy on top of Muffin.[32]

We use Sun's NetBeans 6.1[33] Integrated Development Environment (IDE) to help automatically generate Web services interfaces from created Java code. Each presentation object is implemented as a Web service-equipped resource; and each module in our content adaptation manager is implemented as a Web service-equipped resource as well. Therefore, the interactions between objects and modules are formalized as Web services

communication. Each module can be reused for other purpose; and the integration between modules was smooth.

6.2. *Experiments and discussions*

We have designed a qualitative experiment to evaluate users' satisfaction levels regarding the adapted contents based on our SOA-based content delivery mode. The motivation of this experiment is that we realize content adaptation might result in the degradation of content quality, which may affect people's satisfaction on the adapted content. The reasons of quality degrade might include: reading difficulties due to either a device's screen size or improper content resizing or content reorganization, degrade in fidelity due to either devices' computing and bandwidth limitation or improper trans-coding, and so on. Thus, we designed four evaluation questions to evaluate testers' satisfactions regarding the adapted images, video clips, texts, and page navigation appeared on adapted Web pages. The four questions are as follows:

1. Are testers satisfied with the text browsing appeared on the adapted Web pages?
2. Are testers satisfied with the image browsing appeared on the adapted Web pages?
3. Are testers satisfied with the video browsing appeared on the adapted Web pages?
4. Are testers satisfied with the page navigation after the Web page adaptation?

We choose Yahoo (http://www.yahoo.com) as our content provider and test bed. The reasons why we chose this commonly accessed Web site are as follows. First, Yahoo's site is well known and its overall structure (i.e., its division of sections) is stable. Therefore, our tests can be easily repeated by third parties with similar results. Second, Yahoo's site comprises a comprehensive set of sections, each containing significantly different content types with proprietary static attributes and dynamic features. Therefore, our experiments and results will be more useful and convincing.

This experiment was designed to measure users' satisfaction when they browse different types (i.e., sections or categories) of Yahoo Web pages. Without losing generality, we selected five sections of Yahoo pages, namely *Home*, *News*, *Astrology*, *Auto*, and *Movie*. Each section exhibits some specific features. For example, the *Home* section maintains a relatively stable structure; the *News* section is under frequent updates so that its structure is volatile; the *Astrology*, *Auto*, and *Movie* sections contain a lot of images, photos, audio, and video clips. In short, we used Yahoo's diversified sections as the test bed to evaluate the effectiveness and efficiency of our content adaptation planner.

43 randomly selected testers participate in this evaluation. Each of them was asked to use PDAs and mobile phones to access the aforementioned five sections of the target Web page (Yahoo's *homepage*, *news*, *astrology*, *auto*, and *movie*) after we implemented the content adaptation to the target Yahoo's pages, and then record their answers to the four questions above. Based on their responses, we summarize our findings as shown in Table 1. When testers use PDAs, they are most satisfied with the quality of adapted content appeared at *homepage* section, this is mainly because the *homepage* section has the most stable Web page structure. The testers are least satisfied with the quality of

adapted content appeared at *news* section, this is mainly because the *news* section is updated most frequently among the five section evaluated in this experiment. Among the four types of content been browsed, the testers are most satisfied with the text and page navigation with the satisfaction degrees 96% and 94%, respectively.

However, when the testers are with mobile phones, they oftentimes have difficulties in browsing content due to the smaller screen size and less computing capability, especially in browsing adapted image and adapted video on the *news* section due to its nature of frequent change. (On the *news* section, only 47% and 43% of the testers are satisfied with the quality of *images* and *video*, respectively, which are the two lowest satisfactions in this evaluation experiment).

The testers also expressed their strong demands of content adaptation when they browsed those image-oriented Web pages such as the sections of *astrology*, *auto*, and *movie*. They indicated that a flexible and readable content presentation will be one of their major concerns when they use handheld devices for mobile Internet surfing.

Table 1. Evaluation results of satisfaction of the adapted content on five Yahoo Web sections.

Yahoo section	Types of content been browsed	Satisfaction degree with PDA (%)	Satisfaction degree with phone (%)
homepage	images	83%	67%
	Video	80%	56%
	text	**96%**	87%
	page navigation	**94%**	85%
news	images	68%	**47%**
	video	63%	**43%**
	text	75%	78%
	page navigation	71%	72%
astrology	images	73%	56%
	video	68%	53%
	text	90%	80%
	page navigation	92%	73%
auto	images	79%	57%
	video	75%	51%
	text	87%	82%
	page navigation	89%	74%
movie	images	80%	60%
	video	81%	55%
	text	86%	81%
	page navigation	85%	77%

7. Conclusions

Based on the experimental results, we conclude that our SOA-oriented content delivery model helps to attain the goal of presentation optimization by using minimal information to represent maximized meaning of content. In this paper, we have presented our content

adaptation technique and applied it to improve mobile Internet navigation. Without updating any previous Web content designed for desktop computers, our solution realizes a "One-for-All" feature. The myth of "Write Once, Show Everywhere" allows content providers to prepare a content page only once in HTML oriented to desktop computers; and it can be presented onto various devices with the help of our underlying content adaptation technique. Our experimental results show that our SOA-oriented content delivery model can dramatically improve users' satisfactions of the adapted Web content, especially when people are using handheld devices. We believe that when portable devices are more popular in Web browsing, our content adaptation will have more impacts on the enhancement of mobile Internet navigation.

Although our work was conducted on a specific web site (Yahoo), it can be applied to any other platforms. In general, our SOA-oriented content delivery model can be implemented either as an individual Web service or a plug-in to a Web browser. In other words, it can be used either as an intermediate server or a downloadable on client side.

Acknowledgments

This work is supported by National Science Council, Taiwan under grant NSC95-2520-S-008-006-MY3 and NSC 96-2628-S-008-008-MY3.

References

1. O. Takeshi, C. Simon, A. Nader, and T. Marcus (2002). An Intelligent System for Managing and Utilizing Information Resource over the Internet. *International Journal on Artificial Intelligence Tools (IJAIT)*. Vol. 11, No. 1, pp. 117-138.
2. F.J. Gonzalez-Castano, J. Garcıa-Reinoso, F. Gil-Castineira, E. Costa-Montenegro, and J.M. Pousada-Carballo (2005). Bluetooth-Assisted Context-Awareness in Educational Data Networks. *Computers & Education*. Vol. 45, pp. 105-121.
3. M. Roman, N. Islam, and S. Shoaib (2005). A Wireless Web for Creating and Sharing Personal Content through Handsets, *IEEE Pervasive Computing*.Vol. 4, No. 2, pp. 67-73.
4. M. Satyanarayanan (2004). The Many Faces of Adaptation. *IEEE Pervasive Computing*. Vol. 3, No. 3, pp. 4-5.
5. B.N. Schilit, N.I. Adams, and R. Want (1994). Context-Aware Computing Applications *in Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, USA, pp. 85-90.
6. S.J.H. Yang, J. Zhang, and I.Y.L. Chen (2007). A JESS Enabled Context Elicitation System for Providing Context-Aware Web Services. *Expert Systems with Applications*.
7. S.J.H. Yang and N.W.Y. Shao (2006). An Ontology Based Content Model for Intelligent Web Content Access Services. *International Journal of Web Service Research* (*JWSR*). Vol. 3, No. 2, pp. 59-78.
8. S.J.H. Yang, J. Zhang, R.C.S. Chen, and N.W.Y. Shao (2007). A UOI-based Content Adaptation Method for Improving Web Content Accessibility in the Mobile Internet. *ETRI Journal*, Vol. 29, No. 6, pp. 794-807.
9. S.J.H. Yang, J. Zhang, and I.Y.L. Chen (2007). Ubiquitous Provision of Context-Aware Web Services. *International Journal of Web Services Research* (*JWSR*), Vol. 4, No. 4, pp. 83-103.
10. L.-J. Zhang, J. Zhang, and H. Cai (2007). Services Computing. *Springer & Tsinghua University Press*.

11. M.T. Chebbine, A. Obaid, S. Chebbine, and R. Johnston (2005). Internet Content Adaptation System for Mobile and Heterogeneous Environment. *in Proceedings of Second IFIP International Conference on Wireless and Optical Communications Networks 2005* (*WOCN 2005*), Mar. 6-8, 2005, Dubai, United Arab Emirates, pp. 346-350.
12. Oracle, "Oracle Application Server Wireless", Available from: http://www.oracle.com/ technology/ tech/wireless/index.html.
13. Sun, "Java System Portal Server Mobile Access", Available from: http://www.sun.com/ software/products/portal_srvr/index.xml.
14. IBM, "WebSphere Transcoding Publisher", Available from: http://www-306.ibm.com/ software/ pervasive/transcoding_publisher/.
15. T. Phan, G. Zorpas, and R. Bagrodia (2002). An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients *in Proceedings of the 22nd International Conference on Distributed Computing Systems 2002*, pp. 507-516.
16. G. Berhe, L. Brunie, and J.M. Pierson (2004). Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing *in Proceedings of the First Conference on Computing Frontiers on Computing Frontiers 2004*, pp. 60-69.
17. Y.W. Lee, G. Chandranmenon, and S.C. Miller (2003). GAMMA: A Content Adaptation Server for Wireless Multimedia Applications. *Bell-Labs*, Technical Report.
18. T. Lemlouma and N. Layaida (2004). Context-Aware Adaptation for Mobile Devices *in Proceedings of 2004 IEEE International Conference on Mobile Data Management 2004*, pp. 106-111.
19. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal (1996). Pattern-Oriented Software Architecture: A System of Patterns (POSA). *John Wiley & Sons*, Hoboken, NJ, USA, 1996.
20. R. Arash, B.H. Lawrence, and J.C. Diane (2004). Structural Web Search Engine. *International Journal on Artificial Intelligence Tools* (*IJAIT*), Vol. 13, No. 1, pp. 27-44.
21. VCDGear, Available from: http://www.vcdgear.com.
22. PictView, Available from: http://www.pictview.com/pvw.htm.
23. JafSoft, Available from: http://www.jafsoft.com/asctohtm/.
24. Microsoft, "Microsoft Reader", Available from: http://www.microsoft.com.
25. A. Kinno, H. Yukitomo, and T. Nakayama (2004). An Efficient Caching Mechanism for XML Content Adaptation *in Proceedings of the 10th International Multimedia Modeling Conference* (*MMM*), Jan. 5-7, 2004, pp. 308-315.
26. S.J.H. Yang, J.J.P. Tsai, and C.C. Chen (2003). Fuzzy Rule Base Systems Verification Using High Level Petri Nets. *IEEE Transactions on Knowledge and Data Engineering*, 2003, Vol. 15, No. 2, pp. 457-473.
27. IBM, "Web Services Resource Framework (WSRF)", Available from: http://www-128.ibm.com/developerworks/library/specification/ws-resource/.
28. Z. Uwe and A. Paris (2008). A Catalog of Architectural Primitives for Modeling Architectural Patterns. *Information and Software Technology*. Vol. 50, No. 9-10, pp. 1003-1034.
29. BPEL, "Business Process Execution Language for Web Services (BPEL), Version 1.1." Available from: http://www.ibm.com/developerworks/library/ws-bpel.
30. P. Jyotishman, B. Samik, L. Robyn, and H. Vasant (2008). MoSCoE: An Approach for Composing Web Services Through Iterative Reformulation of Functional Specifications. *International Journal on Artificial Intelligence Tools* (*IJAIT*). Vol. 17, No. 1, pp. 109-138.
31. Muffin, "MUFFIN.DOIT.ORG", Available from: http://muffin.doit.org.
32. Muffin, "Muffin", Available from: http://muffin.doit.org.
33. Sun, "NetBeans", Available from: http://www.netbeans.org/.