A Service Supporting Universal Access to Mobile Internet with Unit of Information-Based Intelligent Content Adaptation

Stephen J.H. Yang[1], Jia Zhang[2], Norman W.Y. Shao[3], Rick C.S. Chen[4]
[1,4]National Central University, Taiwan, [2]Northern Illinois University, USA
[3]Naval Shipbuilding Development Center, Taiwan
[1]jhyang@csie.ncu.edu.tw, [2]jiazhang@cs.niu.edu, [3]snorman@giga.net.tw, [4]chungshiuan@csie.ncu.edu.tw

Abstract

In the mobile Internet, users mostly work with handheld devices with limited computing power and small screens. Their access conditions also change more frequently. In this paper, we present a novel service supporting intelligent content adaptation to better suit handheld devices. The underlying technique is 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. Experimental results demonstrate that our method enables more exquisite content adaptation.

1. Introduction

In a mobile Internet environment, users oftentimes work with handheld devices, such as Personal Digital Assistants (PDAs) and mobile phones, which are featured with good mobility but limited computational capabilities and display sizes. Since most of the existing Web contents are originally designed for display on desktop computers, direct content delivery without layout adjustment and content adaptation often leads to disorganized information on handheld screens. In addition, not every handheld device can play all media types, e.g., a non-multimedia mobile phone cannot play continuous video clips. Furthermore, users' access conditions change more frequently in a mobile Internet environment than in a desktop-based Internet environment [1, 2]. Under some circumstances, it may be unnecessary to deliver all rich media information. For example, if a user is driving, it is unnecessary to deliver video clips because drivers are not supposed to watch video while driving.

Content adaptation thus refers to a technique of dynamically adjusting content presentation to meet the constraints of different receiving devices for better presentation [3]. It can further save unnecessary network bandwidth by removing media that are un-playable on certain devices. The conventional approach of providing different versions of Web content to support various types of receiving devices is not only labor-intensive but also error-prone. Simply changing a multi-column layout to a single-column one also introduces problems with semantics missing among components. As a result, tools and mechanisms are urgently needed to provide seamless Web content adaptation and delivery service to handheld derives.

In the mobile Internet, content adaptation aims to bridge the gap between content providers and mobile consumers. The following four major research issues have to be tackled: (1) representation and detection of mobile user contexts, (2) adaptation rule design, (3) content adaptation management, and (4) format (e.g., stylesheet) generation. In this paper, we will focus on problem (3): content adaptation.

A digital content is typically composed of multimedia objects such as text, images, audio and video. These objects are connected with each other via different relationships, e.g., an image can illustrate a section of text article; a text title can abstract a text article or some images. In other words, these related objects are synergistically integrated to help readers understand what authors intend to express. Improper rearrangement of these objects and their relationships may lead to ambiguous expression or loss of information. Therefore, it is important for a content adaptation mechanism to maintain the original semantic consistence among comprising objects during an adaptation process.

In this paper, we present a novel service supporting dynamic Unit of Information (UOI)-based content adaptation for handheld devices. Our goal is to improve Web content accessibility in the mobile Internet, while retaining semantic coherence of the original contents. To achieve this goal, we introduce UOI as an atomic presentation unit of a Web page; all media objects in a UOI have to be presented as a whole. We present an algorithm to automatically identify and detect UOIs from Web pages. Experiments showed that our UOI detection algorithm successfully identified 78% of UOI segments in our test bed. We also found that our method performs well for well-formatted Web pages.

The remainder of this paper is organized as follows. We first introduce related work in Section 2. Then we present our dynamic content adaptation framework, information fragment detection mechanism, and content adaptation method in Section 3. Afterwards, we present our experimental designs and result analyses in Section 4. Finally, we draw conclusions in Section 5.

2. Related work

Associating with Extensible Stylesheet Language (XSL), XML-based transformation has been widely used to support heterogeneous content adaptation [4-6]. XSL Transformation (XSLT) is usually used as a generic XML transformation engine, by transforming one format of structured XML data into another.

Many content adaptation prototypes have been built in recent years. Among them, Phan et al. [6] propose a middleware, called Content Adaptation Pipeline (CAP), to perform content adaptation on any complex data types, not only text and graphic images. XML is used to describe all the elements in a content hierarchy. Berhe et al. [7] 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. A logic adaptation path is determined by associating adaptation constraints to proper combinations of adaptation operators. To determine the optimal service, a path selection algorithm is proposed based on cost and time considerations. Their work shows a proof-of-concept of Web-based content adaptation; however, the implementation is still in a preliminary phase. How to map from constraints to adaptation operators is unsolved. Lee et al. [8] 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 [5] propose an adaptation framework, which defines an adaptation strategy as a set of description models, communication protocols, and negotiation and adaptation methods. Two adaptation methods are examined: XSLT-based structural adaptation and media adaptation [9].

Some researchers focus on content decomposition methods. Chen et al. [10] propose a block-based content decomposition method, DRESS, for quantifying the content representation. An HTML page is factorized into blocks, each being assigned a score denoting its significance. DRESS selects the block with the highest score to represent the content. This method prevents from missing significant information. It also enables content layout to become adjustable according to the region of interest, attention value, and minimum perceptible size [11]. Ramaswamy et al. [12] propose an efficient fragment generation and caching method based on detection of three features: shared behavior, lifetime, and personalization characteristic. The smallest adjustable element in these two approaches is a composite of objects (i.e., text, image, audio, and video). This granularity of decomposition is too large for mobile device screens; therefore, they are not suitable for mobile content adaptation.

Our previous studies in content adaptation [13, 14] focus on multi-column to single-column layout transformation. We have proved that this method can provide better browsing experience for mobile devices. However, we found some semantic errors appear when adjacent media objects crosscut. These errors may confuse users. To overcome this deficiency, in this paper we introduce the concept of UOI and present an algorithm to automatically identify semantically coherent presentation unit of components that have to be shown together.

3. Content adaptation service

We implemented a Content Adaptation Engine (CAE) to provide intelligent content adaptation Web service for mobile devices. As illustrated in Figure 1, CAE contains three major components for a consecutive process of transforming original contents into adapted ones: decomposition, transformation, and composition. In the decomposition phase, the original Web page is structurally parsed into components based on predefined content model. Both the layout and constituent elements (e.g., text, image, audio, and video) are extracted separately from this phase. In the transformation phase, transcoding approaches are used to change the fidelity and/or modality of the extracted components for better representation on target devices. In the composition phase, the presentation styles (layouts) and the adapted components are reorganized and recomposed into the final contents to be delivered to the end users.

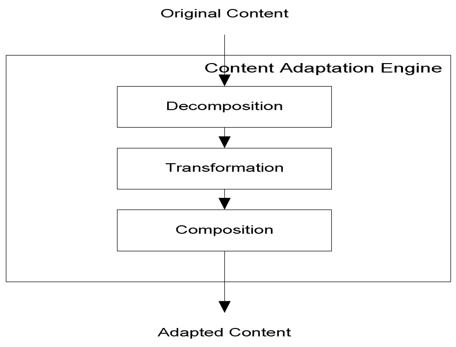

Figure 1. Three phases of content adaptation.

The interface of CAE is shown as follows in WSDL, by providing one *portType* with one *operation* “adaptContent.” The operation accepts a SOAP message “adaptContentRequest” and replies with a SOAP message “adaptContentResponse.”

```
<portType name="adaptContent">
  <operation name="adaptContent">
    <input message="adaptContentRequest"/>
    <output message="adaptContentResponse"/>
  </operation>
</portType>
```

3.1 Content Structure Model (CSM)

A Web page typically contains a set of media objects carrying encapsulated meanings. The semantics among presentation components have to be maintained to deliver correct information. For example, an illustrative figure should be shown close to its detailed text message. When some contents are adapted to be displayed onto different devices, the semantics of the decomposed portions in the adapted contents should remain the same as those in the original contents. In other words, adapted objects should be grouped based on semantic consistency. As a result, how to determine object grouping is the most critical step.

We formalize this object grouping requirement into an isomorphism problem: the relationships among objects and formed groups before and after adaptation should be able to be expressed by an isomorphic graph. To solve this problem, we utilize a layered Content Structure Model (CSM) [13] to organize objects with possible presentation versions of a given Web page. As shown in Figure 2, a CSM maintains available adaptation rules and possibilities for individual presentation objects. According to CSM, Web content is organized in 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. For example, object *OC6* in Figure 2 may be presented in four presentation types: *video*, *audio*, *text*, and *image*. Its audio presentation type can be provided in three formats: *mp3*, *wmv*, and *midi*. If the end user is using an mp3 player while driving, *OC6* should be provided in *audio* using an *mp3* format.

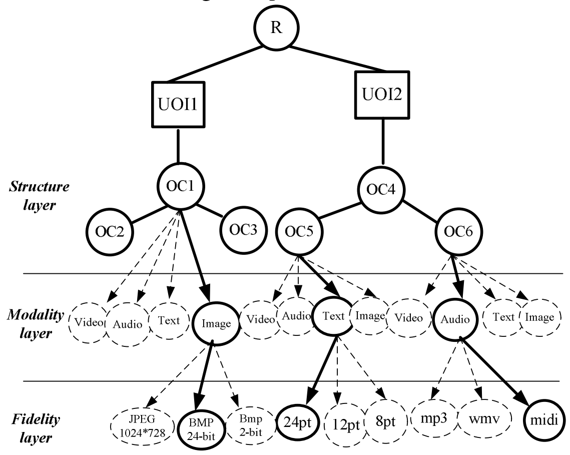

Figure 2. Content Structure Model (CSM).

We then extend CSM by incorporating object relations into its structure layer. The goal is to maintain semantic inherences among objects in layout re-arrangement to enable more exquisite content adaptation under various circumstances and contexts.

3.2 Unit of Information (UOI)

As shown in Figure 2, we define an atomic information unit, or so-called Unit of Information (UOI), as a semantic unit comprising a set of segments and media objects that have to be presented together on the same screen. In our research, UOI is considered as the basic presentation unit of Web content. In other words, a Web page can be presented by a composition of multiple UOIs. In CSM, composition of UOIs is an expression in the structure layer. The aim of UOI is to preserve the semantic coherence of Web contents throughout adaptation processes. UoIs have to be identified in the decomposition phase; the subsequent transformation and composition phases have to retain the UoIs unbroken.

A UOI contains two types of elements: segments and object clusters. To design a Web page content in a markup language (e.g., HTML), authors typically use various partition elements (e.g., HTML tags such as <frameset>, <table>, and <div>) to arrange the layout of information objects. These partition elements contain no substantial information but layout arrangements and containing relationships. Each of these partition elements is called a segment. Thus, a Web page can be decomposed into a number of segments organized in a hierarchical structure, as shown in Figure 3.

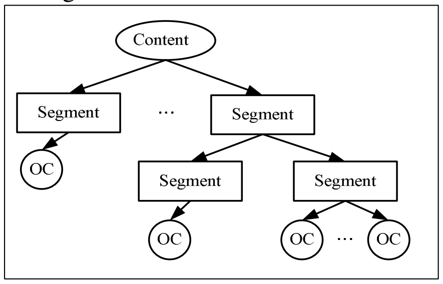

Figure 3. The relationship among content, segment, and object cluster (OC).

Segments can be further classified into two types: arranging segment (AS) and containing segment (CS). Figure 4 illustrates the concepts and relationships between them. An AS refers to a partition element that contains no concrete media objects as direct children. It is used to define the layout of a specific portion of a Web page. In contrast, a CS refers to a partition element that contains at least one concrete media object as a child.

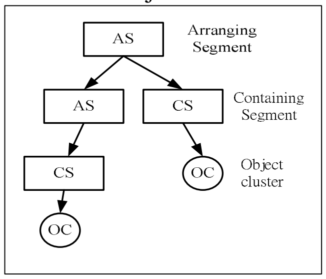

Figure 4. Arranging segment and containing segment (on a

segment tree).

All media objects of a Web page are further classified into different object clusters based on their types. Without losing generality, in this research, we consider four types: text, image, audio, and video. After a parsing process, the presentation components are identified as "objects" associating with presentation attributes. The objects with the same attributes (i.e., modality) may have the same semantic hierarchies. An object cluster is thus defined as a collection of media objects that possess the same modality inside of the same containing segment (CS). Six types of object clusters are identified: (1) text cluster (TC) (e.g., text), (2) still image cluster (SIC) (e.g., jpg, bmp, tiff, and gif objects), (3) video cluster (VC) (e.g., avi, wmv, and mpg objects), (4) dynamic image cluster (DIC) (e.g., png and gif objects), (5) flash cluster (FC) (e.g., swf objects), and (6) audio cluster (AC) (e.g., mp3 and wav objects).

3.3 Construction of segment trees

We define an intermediate segment tree as a presentation model, which contains segments and object clusters of a Web page and organizes them in a tree-like structure (as show in Figure 4). The purpose of constructing an intermediate segment tree is for detecting UOIs in a Web page. The pseudo code of constructing a segment tree is shown in Figure 5.

```
page = cleanup(page);
tree = parsePage(page);

//Annotate object clusters
public void markOC(Node n) {
  if (n.children != null) {
      for (int i = 0; i <= n.children.length(); i++)
        markOC(n.children[i]);
  }
if ((n.isTC() || n.isSIC() || n.isVC() || n.isDIC() ||
n.isFC() || n.isAC())
  n.type = "OC";
}

//Annotate AS & CS
public void markASCS(Node n) {
  if (n.children != null) {
    for (int i = 0; i <= n.children.length(); i++)
      markASCS(n.children[i]);
    }
  if ((n.type!="OC")&&(n.numOfOCChildren()>=1))
    n.type = "CS";
  else n.type = "AS";
}
```

Figure 5. Algorithm of constructing segment tree.

HTML provides great flexibility to integrate a variety of multimedia types; however, it allows free style writing that makes it hard to identify and determine various types of objects in an HTML document. To overcome this problem, our first step is to transform the content into a well-formed format. We adopted an open-source package "HTML tidy" [15] for conducting the task.

Then the cleaned-up HTML page is parsed into a tree-like structure, each node representing a tag in the page. In theory, any XML parser can be used to parse the HTML content. The generated tree structure is traversed for searching object clusters. According to the six types of object clusters, we use file extensions to identify object clusters. Take the following tag as an example:

"<img src=http://news/peace.jpg width="50">"

The tag node is considered as a still image cluster (SIC) due to its file extension "jpg." In general, any tree traversal algorithm is applicable here. We adopted recursive post-order traversal algorithm, where each node is visited after all of its children nodes are visited.

After all object clusters are annotated, we traverse the segment tree once again to identify containing segments and arranging segments. Take the following example:

<li id=" 82"> <a href= 27> <img
src="http://news/peace.jpg" width="" height="21">
Holiday wreath sparks controversy
</a></li>

Recall that the "image" segment has been annotated as an object cluster. Its enclosing segment "a href" contains an object cluster; therefore, it is marked as a containing segment. Since the outmost segment "li" only has one containing segment as a direct child, it is marked as an arranging segment.

The result of this process is a segment tree, each node being annotated as one of the three categories: OC, CS, or AS. The next step is to identify and detect UOIs in the segment tree.

3.4 Identification and detection of UOIs

Figure 6 shows our UOI detection algorithm that is designed based on a constructed segment tree. To ease explanation, Figure 7 illustrates the detailed rules of how to annotate and merge various segment nodes to identify UOIs. Our algorithm goes through a two-phase process: the first phase traverses the initial segment tree and annotates an initial set of UOIs (Step 1); the second phase traverses the result segment tree from phase 1 to further identify all possible UOIs (Step 2 ~ Step 4).

```
//Step 1. Annotate UOIs (post-order)
public void markUOI(Node n) {
  if (n.children != null) {
    for (int i = 0; i <= n.children.length(); i++)
      markUOI(n.children[i]);
  }
if ((n.type =="as") && (n.color != null) &&
(n.numOfOCChildren() >=2))
```

```
  n.type = “uoi”;
 }

// Step 2. Identify UOI candidates and groups
public void markUOICandidateGroup(Node n) {
  if (n.type == “uoi”) return;   //uoi already
  if (n.children != null) {
    for (int i = 0; i <= n.children.length(); i++)
      markUOICandidateGroup(n.children[i]);
  }
  if ((n.type == “cs”) && (n.numOfOCChildren() >= 2))
    n.type = “uoic”;
  if ((n.type==“cs”) && (n.numOfOCChildren()==1))
    n.type = “group”;
  }

// Step 3. UOI determination
public int determineUOI (Node n) {
  if (n.children != null)
    for (int i = 0; i <= n.children.length(); i++)
      determineUOI (n.children[i]);

  //3.0 if all children are UOI candidates
  if (n.children != null) {
    boolean flag = true;
    for (int i = 0; i <= n.child.length(); i++)
    if (n.child[i].type != “uoic”) {flag = false; break;}
    if (flag) n.type = “uoic”;
    return 0;
  }
  if (n.type != “group”) return 0;

  //3.1 merge group with UOI candidate child
  if (n.contain_UOIC_child())  n.type = “uoic”;

  //3.2 merge group with adjacent UOI candidate
  if ((!contain_uoic_child()) && (n.has_uoic_sibling()))
    n.type = “uoic”;

  //3.3 merge group with adjacent group
  if ((!contain_uoic_child()) && (!n.has_uoic_sibling())
&& (!n.has_group_sibling())) {
    n.type = “uoic”;
    //assign “uoic” to adjacent group
  }

  //3.4 merge group upward
  if ((n.child == null) && (!n.has_sibling())) {
    n.parent = “group”;
    return “-1”;
  }
}

// Step 4. UOI Remark
public void remarkUOI(Node n) {
```

```
  if (!groupExist(n)) {
    if (n.children != null)
      for (int i = 0; i <= n.children.length(); i++)
        markUOICandidateGroup(n.children[i]);
    if (n.type == “uoic”) n.type = “uoi”;
  }
}
```

Figure 6. UOI detection. algorithm

In Step 1, the initial segment tree is recursively traversed in post-order to identify all UOIs. As shown in Figure 7(1), a segment node is annotated as a UOI if it meets all three conditions: (1) its type is AS, (2) it has been annotated with color attribute, and (3) it contains at least two OC children.

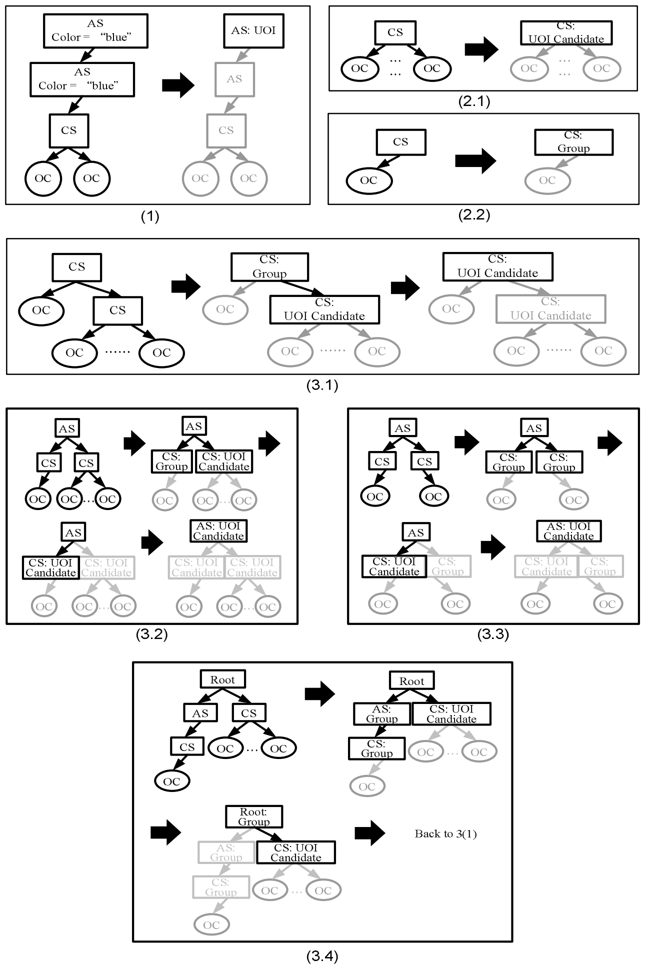

Figure 7. UOI identification and detection process.

Step 2 intends to identify UOI candidates and Groups in a segment tree. As shown in Figure 7(2.1), a segment is marked as a UOI candidate if it meets two conditions: (1) the segment type is CS; and (2) the segment contains at least two OC children. As shown in Figure 7(2.2), a segment is marked as a Group if it meets two conditions:

(1) the segment type is CS; and (2) the segment contains only one OC child.

Step 3 deduces more UOIs by merging UOI candidates and Groups in the result segment tree in four ways. In Step 3.1, as shown in Figure 7(3.1), if a Group contains a UOI candidate as a child, it merges with its UOI candidate child to form a new UOI candidate. In Step 3.2, as shown in Figure 7(3.2), if a Group contains no UOI candidate children but has an adjacent UOI candidate sibling, it merges with the UOI candidate sibling to form a new UOI candidate. If the newly-formed UOI candidate has no siblings, it is further merged with its parent to form a new UOI candidate. In Step 3.3, as shown in Figure 7(3.3), if a Group has neither UOI candidate children nor siblings but has an adjacent Group sibling, it merges with its adjacent Group sibling to form a new UOI candidate. If the new-formed UOI candidate has no siblings, it is further merged with its parent to form another UOI candidate. In Step 3.4, as shown in Figure 7(3.4), if a Group does not have any child or sibling, it is merged with its parent to form a new Group, and the process goes back to Step 3.1.

Finally, Step 4 cleans up the resulting segment tree. If no Group exists in the segment tree, all UOI candidates are marked as UOIs.

3.5 Content adaptation for mobile devices

The aforementioned algorithm helps to automatically detect all UOIs of a Web page. Through this process, a segment tree (formatted content tree) is transformed into a CSM tree annotated by UOIs, which can be used to generate an adaptation tree leading to the final adapted content (e.g., in HTML format) based on delivery contexts. Figure 8 illustrates the relationships between a content tree, its CSM tree, its adaptation tree, as well as example adaptation rules.

Figure 8(a) shows the original content designed for PC or Notebook (NB). To browse the same content via a PDA, however, its size is far larger than a PDA's screen as shown on Figure 8(b). Therefore, the content tree (in HTML) generated from the original content is translated into an object relation tree, by extracting the attributes of the objects and segments. Based on these attributes, we can construct a CSM tree.

Afterwards, if each UOI fits on the PDA's screen, then their locations are rearranged through column-oriented rearrangement, as shown in Figure 8(c) [13, 14].

If the largest size of UOIs cannot fit into a small-size screen like a wireless phone, the scales and positions of the objects in the UOI should be further adjusted, as shown in Figure 8(d). This paper focuses on UOI-based fragment detection; the algorithm of adjusting layout positions will not be discussed.

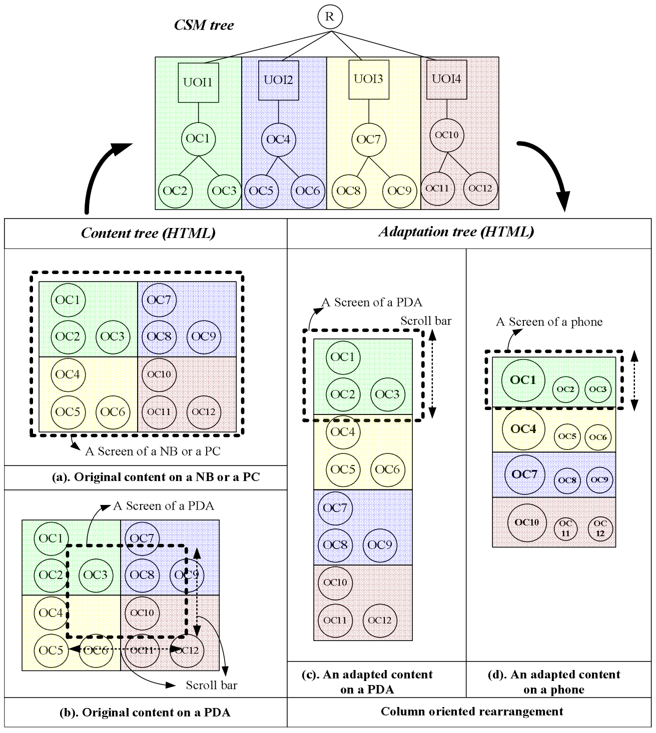

Figure 8. Content adaptation process.

4. Experiments

To evaluate our UOI detection algorithm, we designed and conducted a set of experiments to examine visual performances and conduct quantitative analysis. A set of 35 Web sites are selected and grouped into four categories: 5 from academia, 2 from news, 11 from business corporations, and 17 from general Web portals. Without losing generality, the selection of the Web pages as testing samples is generally random-based and depends on statistics of students' daily access. For each Web site, we used a PDA screen to visualize three results: (1) the Web page without any adaptation, (2) the Web page after standalone page adaptation algorithm (transforming multi-column to single-column layout), and (3) the Web page after UOI detection-based page adaptation. The visualization results for each Web site under the three strategies were captured as screen shots for comparison.

To perform quantitative analysis, we designed a four-stage validation procedure to measure the correctness rate of our UOI detection algorithm on each of the 35 test cases. Stage 1 builds a target baseline. A Web page is manually browsed to identify all UOIs using common sense. The information objects, which have the same or similar semantic meanings, are grouped as a UOI. Stage 2 executes our algorithm. The identification process is automatically executed and monitored. Stage 3 evaluates the results against the baseline. The results from Stage 1 and 2 are compared to calculate the correctness rate of our UOI detection algorithm. Stage 4 visually validates whether our algorithm could maintain the semantic meanings of the original contents. The method is to adapt the Web page based on identified UOIs from Stage 2 and show the results on a PDA screen. The presentation sequences of the page is rearranged by forming a single-column layout based on identified UOIs. Note that this is just one simple yet efficient way to evaluate our algorithm. The visual effect is examined to validate whether the original semantic meanings are maintained.

4.1 Analysis on visual effects

Figure 9 shows the visualized results on a PDA using the three strategies for three randomly selected Web sites: Inaba, B'z, and Yahoo. The results for each Web site occupy one row, which comprises three screen shots: original content, adapted content without UOI detection, and adapted content based on UOI detection. As shown in Figure 9, content adaptation based on our UOI detection algorithm effectively reorganizes and adjusts the original content on handheld screens.

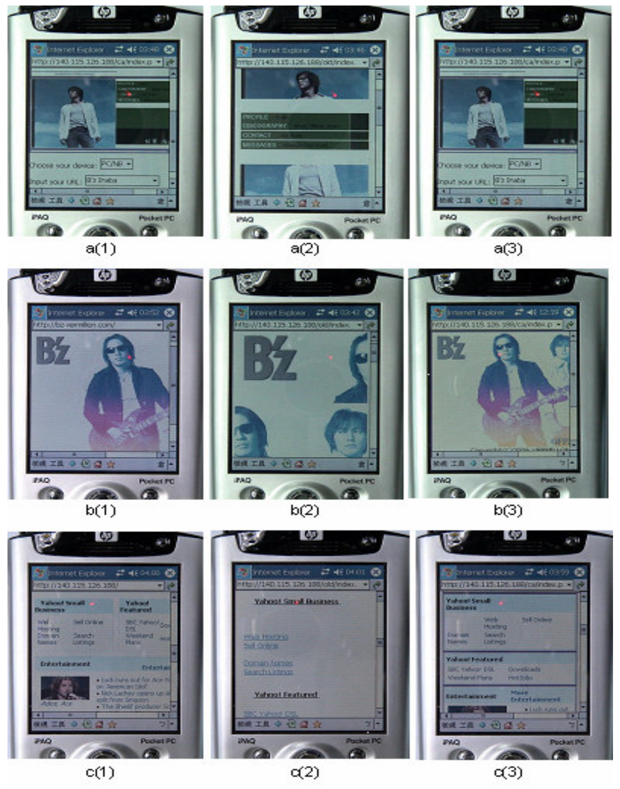

Figure 9. Comparison of visualization effects.

As shown in Figure 9, using the standalone transformation approach, each object is treated individually and independently, and the adaptation process may scale up one image object to the entire screen size. As a result, some unrelated information objects may be inserted into these related image objects, as shown in the 2nd column in Figure 9. This simple object-based adaptation may cause confusing representation. In contrast, by applying our UOI concept, the original semantic meaning associated with objects can be preserved in the process of content adaptation according to the delivery context.

4.2 Statistical analysis on targeted Web pages

We conducted the tests over the selected 35 testing samples. For each testing Web page, we measured the results in the following five factors: (1) the number of manually identified UOIs, (2) errors occurred in the decomposition phase, (3) errors occurred in the composition phase, (4) statistics of incorrectly identified UOIs, and (5) statistics of correctly identified UOIs. The detected errors caused by the UOI detection algorithm in Stage 3 is counted and analyzed in the decomposition phase; the detected errors caused by the presentation rearrangement in Stage 4 is counted and analyzed in the composition phase. Errors caused by decomposition are further divided into two categories: data loss in the pre-process and errors from UOI detection algorithm. Errors caused by composition are further divided into two categories: errors cause by mis-arrangement and errors caused by composition.

According to the test results, our UOI detection algorithm successfully detected 78.49% of UOIs as an average. In more detail, our algorithm has the highest correctness rate for the academic pages (88.06%) and the business (88.04%) pages, then the general portals (72.98%), and the lowest correctness rate for the news pages (48.89%). The data also show that our algorithm has promising correctness rate of UOI detection with well-formatted Web pages. By examining and monitoring one business homepage IBM and one news Web site CNN with low correctness rate, we found that these sites undergo frequent changes. We further found that these pages are designed comprising enormous number of "table" fragments and much multimedia information. This causes our algorithm produce many false segment nodes and thus leads to mis-merging in Step 3.

4.3 Analysis on detection errors

We further investigated the found UOI detection errors for each Web site and categorized them into different causes. Based on the statistical information, a pie chart is used to summarize and illustrate the different causes. As shown in Figure 10, the percentage of detection errors due to the Step 3 of the UOI detection algorithm (in Figure 6) is 70.72%; due to information lost after composition is 17.13%; due to mis-arrangement is 7.18%, and due to data loss in the pre-process is 4.97%.

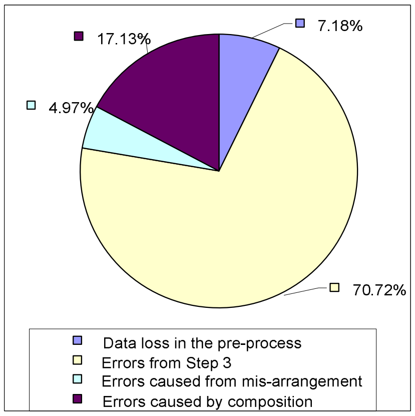

Figure 10. The distribution of error categories.

We found that some information objects may get lost after the format transformation in the decomposition phase. We also found that our merging algorithm needs further enhancement when corresponding HTML scripts do not follow exact formats.

Transforming a multi-column layout into a single-column layout caused two kinds of errors in our experiments: one is from the composition operation; the other is from mis-arrangement.

5. Conclusions

In this paper, we have presented a UOI-based dynamic content adaptation approach. We present algorithms that automatically detect semantic relationships between comprising components in Web content, and then reorganize page layout to suit handheld devices based on identified UOIs. Our experiments have proved that our UOI detection algorithm: (1) is effective on preserving semantic meanings and coherence of information objects in a Web page, and (2) can largely help and facilitate in Web page adaptation to mobile devices. Our experiments also show that our UOI detection algorithm works well with well-formatted Web pages. Regarding ill-formatted Web pages, it seems that more cleanup work is necessary in addition to the Tidy package we adopted. This work can be conducted in our future research.

We also found several barriers existing in the current adaptation techniques, such as lacking the capability of processing script languages (e.g., JavaScript and VBScript), and lacking session and message processing mechanism (e.g. login session).

We found an interesting phenomenon, that many Web pages have similar layout structures but small differences, even though their contents are significantly different. These similar but rarely changed portions (e.g., header fragment and navigation fragment) occupy significant storage space and consume many computing resources. To speed up the decomposing time and reduce the required storage space, we will continue to work on intelligent fragment detection method to find out similar fragments among Web pages.

6. References

[1] M. Satyanarayanan, "The Many Faces of Adaptation," IEEE Pervasive Computing, Jul.-Sep., 2004, 3(3): pp. 4-5.
[2] M. Roman, N. Islam, and S. Shoaib, "A Wireless Web for Creating and Sharing Personal Content through Handsets," IEEE Pervasive Computing, Jan.-Mar., 2005, 4(2): pp. 67-73.
[3] M.T. Chebbine, A. Obaid, S. Chebbine, and R. Johnston, "*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)*, Dubai, United Arab Emirates, March 6-8, 2005, pp. 346-350.
[4] A. Kinno, H. Yukitomo, and T. Nakayama, "*An Efficient Caching Mechanism for XML Content Adaptation*," in Proceedings of *the 10th International Multimedia Modeling Conference (MMM'04)*, Jan., 2004, pp. 308-315.
[5] T. Lemlouma and N. Layaida, "*Context-Aware Adaptation for Mobile Devices*," in Proceedings of *2004 IEEE International Conference on Mobile Data Management*, 2004, pp. 106-111.
[6] T. Phan, G. Zorpas, and R. Bagrodia, "*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.
[7] G. Berhe, L. Brunie, and J.M. Pierson, "*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.
[8] Y.W. Lee, G. Chandranmenon, and S.C. Miller, *GAMMA: A Content Adaptation Server for Wireless Multimedia Applications*. 2003, Bell-Labs, Technical Report.
[9] Z. Lei and N.D. Georganas, "*Context-based Media Adaptation in Pervasive Computing*," in Proceedings of *Canadian Conference on Electrical and Computer Engineering*, May, 2001.
[10] L.Q. Chen, X. Xie, W.Y. Ma, H.J. Zhang, H.Q. Zhou, and H.Q. Feng, *DRESS: A Slicing Tree Based Web Representation for Various Display Sizes*. 2002a, Technical report MSR-TR-2002-126, Microsoft Research.
[11] L.Q. Chen, X. Xie, X. Fan, W.Y. Ma, H.J. Zhang, and H.Q. Zhou, *A Visual Attention Model for Adapting Images on Small Displays*. 2002b, Technical report MSR-TR-2002-125, Microsoft Research.
[12] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, "Automatic Fragment Detection in Dynamic Web Pages and Its Impact on Caching," IEEE Transactions on Knowledge and Data engineering, 2005, 17(6): pp. 859-874.
[13] S.J.H. Yang and N.W.Y. Shao, "An Ontology Based Content Model for Intelligent Web Content Access Services," International Journal of Web Service Research (JWSR), Apr.-Jun., 2006, 3(2): pp. 59-78.
[14] S.J.H. Yang and N.W.Y. Shao, "Enhancing Pervasive Web Accessibility with Rule-Based Adaptation Strategy," Expert System with Applications, May, 2007, 32(4): pp. 1154-1167.
[15] "Tidy," Available from: http://tidy.sourceforge.net/.