A CLIENT-SERVER MODEL FOR EDITING ODF DOCUMENTS ON MOBILE DEVICES

Zoltán HORVÁTH*, Imre BARNA*, PÉTER BAUER*, Kinga BERNÁD*, Zsolt HERNÁTH**,
Balázs KŐSZEGI*, Gergely KOVÁCS*, Tamás KOZSIK*, Zsolt LENGYEL***,
Róbert ROTH*, Sándor SIKE***, Gábor TAKÁCS*
*Department of Programming Languages and Compilers, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary, e-mail: {hz, bib, bauer_p, bekraai, pma, sanyisd, kto, rorraai, baller}@inf.elte.hu
**Department of Information Systems, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary, e-mail: hernath@inf.elte.hu
***Department of Software Technology and Methodology, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/C, H-1117 Budapest, Hungary, e-mail: {lengyel, sike}@inf.elte.hu

ABSTRACT

Open Document Format (ODF) is a popular office document format accepted by most of modern desktop office suites. The aim of our project is to create a software suite of specialized mobile ODF schemata and editors to provide support of editing ODF-based documents on mobile devices. Number of tests have been carried out with prototype tools focusing on the resource need of transferring, visualizing and editing simple ODF documents. Based on test results we have found that the limited capacity of mobile devices (compared to desktop computers) implies that documents in their original form cannot be handled on mobile devices. In this paper we investigate the methods and background of ODF-based document transfer and edition in a client-server model.

Keywords: *mobile office, mobile software, ODF, XML, RELAX NG, schema, schema transformation*

1. INTRODUCTION

The continuous evolution of mobile devices provides the possibility of implementing mobile software to handle office documents. Open Document Format (ODF) is an open standard and accepted by numerous office suites on desktop computers. The aim of our project is to establish a client-server software model enabling document distribution over computers and mobile devices, and editing office documents on the mobile site.

We implemented special editor prototypes to measure the editing capabilities of a wide range of mobile devices. We generated a variety of documents of different complexity and size, and measured the resource need of different editor operations performed on test documents through a scripting interface [1]. ODF documents' complexity can be characterized by the narrowest schema derivative of ODF schema the document is conformed to.

Prototypes have proved that mobile devices with different capabilities (processor capacity, display size, memory, software platform) are able to handle documents of different complexity and size. This paper focuses on document transformation needed to distribute ODF documents over mobile devices and desktop computers. Distribution includes extracting a part or a whole document from desktop site, transferring extractions to mobile site, edit them there and transfer them back.

ODF as a set of XML documents [7] is described by Open Document Format Specification [2, 3] in terms of a RELAX NG [4–6] schema. The complexity of documents to be transferred to a mobile device can be tailored by transforming the original ODF schema to a derivative, which excludes certain semantic structures. Different derivatives are required for different devices and user requirements. Schema derivatives provide the rules for transforming the original document to the one which on the one hand matches user requirements, on the other hand suits to specific device capabilities.

Since it is not possible that each set of user requirements could meet the capabilities of each unique device, a (relatively small) number of document class, called *profiles*, has been introduced to classify user requirements and device capabilities.

The rest of the paper is organized as follows. In section 2 document distribution is detailed. Section 3 discusses profiles and the profile selection based on information of the mobile device. The method of schema transformation is detailed in section 4. We discuss document transformation in section 5. Our conclusion is given in section 6.

2. DISTRIBUTING ODF DOCUMENTS OVER SERVER AND CLIENT

In our approach editing documents on a mobile platform is achieved by document independent preparations, and handling the document in question.

Document independent preparation to establish proper software environment:

1. *Profile selection:* Mobile users select the proper profile considering the capacity of the mobile device and their document handling's requirements. The decision process can be supported by a simple program that discovers the properties of the given device, and suggests an appropriate profile. The user can override this suggestion risking bad performance for a wider spectrum of features to access. This selection should be made once, before the first use of the mobile editor, and may be overridden later.

2. *Schema derivation:* A schema for each profile defined should be derived by transforming the original ODF schema using a macro language under development. Derived schemata keep conformity against RELAX NG grammar. Schema transformation is

performed only once, and has to be repeated only if the ODF schema or the profile set is changed.

Handling document, an interaction between desktop computer and mobile device:

1. *Document transformation:* When the mobile device connects to the computer and requests a given document, the server site identifies the profile and transforms the document by using the appropriate schema. Transforming the document those parts of the document that are not conform against the schema selected are replaced by unique, typed markers. Replaced parts are stored along with their markers in a separate document on the computer.

2. *Editing on the mobile device:* The transformed document is sent to the mobile device, where it can be viewed and changed by the client program. As markers are typed, they can be displayed as icons if appropriate. During the edition process the document conformity against the derived schema is kept. The latter guarantees that the markers can be replaced later by the original components. The new document is sent back to the computer.

3. *Document reconstruction:* Markers are replaced by the original components and the result is a new document that is valid against the ODF schema.

3. PROFILE SELECTION

Mobile devices perform differently. Some have lots of memory, some have fast processors. Others may lack good hardware, but their optimized software still provide a good performance. All these factors contribute to the overall performance of a device, and none of them may be neglected when estimating the performance of our software with different profiles.

The software platform chosen as the Java Micro Edition that hides low-level hardware information, e.g. the speed of the processor, the amount of internal memory available etc. The capabilities of a mobile device should therefore be estimated by benchmarking the cost of the operations to be performed. Building a resource model [8] consists of three steps:

1. *Identifying schema-related operations:* Different profile-associated schemata provided for the user requirements involve working with different objects like tables, graphics, paragraphs, charts etc. Working with an object means executing some of the basic operations. In each schema derivations we determine the frequency of each basic operation. This has to be done only once for each schema.

2. *Identifying device profile:* Before running the mobile editor for the first time, a series of tests are executed on the mobile device to measure the performance of different basic operations. We have defined the basic operations required for editing documents, and grouped into several categories as follows:

   - *File operations* – reading and writing files from and onto internal memory or memory card, parsing XML, extracting and compressing ZIP files.
   - *String operations* – string concatenation, substring look-up.
   - *Display operations* – displaying various user interface elements.

3. *Offering a schema derivative:* Based on the costs of basic operations and their frequency, we can estimate the total amount of costs concerning a derived schema. By defining a threshold based on the device profile, schema derivatives can be offered for the user. Users have the possibility to override the schema selection manually, but that may cause a lower application performance.

4. SCHEMA TRANSFORMATIONS

Before all, it is important to see that a RELAX NG grammar defines document frames by using patterns `<define...> ... </define>` and inside that frame patterns `<element...> ... </element>`, and `<attribute...>...</attribute>` for the latter also permitting the form `<attribute.../>`. Due to that any schema transformation resulting in some derivative can be composed by a particular sequences of of schema transformation primitives defined as follows:

1. removing `<define...> ... </define>` patterns by name, together with all `<ref.../>` patterns referencing the same name,

2. introducing new `<define...> ... </define>` patterns,

3. removing `<element...> ... </element>` patterns by name, together with all `<ref.../>` patterns referencing the same name,

4. introducing new `<element...> ... </element>` patterns,

5. removing `<attribute...> ... </attribute>` or `<attribute.../>` pattern, together with all `<ref.../>` patterns referencing the same name,

6. introducing new `<attribute...> ... </attribute>` or `<attribute.../>` patterns,

7. removing content-less patterns (like `<choice> </choice>`, etc), occasionally resulted by remove operations numbered 1, 3 and 5.

It is obvious that any schema transformation can be described as a particular sequence of the primitives above. As

previously stated, for each profile there is an associated derived schema. The question is how particular profiles can be achieved for users. One way is to predefine a collection of editor profiles and, for users, the only possibility to select one of them. The disadvantages of this way is that it makes a strong restriction for users. Another way is to allow the user to freely describe a desired schema transformation as a particular sequence of the primitives above. This way seems a better one, but only for those users who are familiar with ODF grammars. An ideal way would be to provide a formal document frame and style description system in strong relationship with schema transformation primitives in which users may and can freely construct their ideal document style and frame. This way is even much rather ideal, since it offer the possibility of computer controlled automatic generation of the adequate macro definitions and embedding guaranteeing derived schema's consistency and conformity.

5. DOCUMENT TRANSFORMATION

A document is transformed before transferring to the mobile device according to the RELAX NG schema relating to the selected profile. Transformation takes the document and the RELAX NG schema as input, and outputs two files:

- *The transformed document* contains typed markers each equipped with a unique ID.
- *Marker definition file* is an XML document containing pairs of markers and XML fragments. When committing mobile site changes, each marker is replaced with the associated XML fragment. Since markers are typed the document conformity against ODF schema is guaranteed.

The transformation is done by validating the document against the schema derivative. Whenever a document fragment is found (a sub-tree in the graph representation of the document) which is not allowed by the derived schema, we place a typed marker with a unique ID in its place while saving the replaced document fragment in the marker definition file associated with the marker. This means we provide a bijection between markers and document fragments.

Here we present a basic example. Let the input document fragment be:

```
...
<table name="FirstTable" style="FirstTable">
  <column style="FirstTable.A" />
  <column style="FirstTable.B" />
  <row style="FirstTable.1">
    <cell style="FirstTable.A1">
      Hello
      <list>
        <item>First </item>
        <item>Second</item>
      </list>
    </cell>
    <cell style="FirstTable.B1">
      World
    </cell>
  </row>
</table>
...
```

Let us suppose that the schema does not allow placing a list inside a table. The transformed document is as follows:

```
...
<table name="FirstTable" style="FirstTable">
  <column style="FirstTable.A" />
  <column style="FirstTable.B" />
  <row style="FirstTable.1">
    <cell style="FirstTable.A1">
      Hello
      <marker type="list" id="0" />
    </cell>
    <cell style="FirstTable.B1">
      World
    </cell>
  </row>
</table>
...
```

The marker definition file:

```
...
<markerDefinition>
  <marker type="list" id="0" />
  <list>
    <item>First </item>
    <item>Second</item>
  </list>
</markerDefinition>
...
```

6. CONCLUSIONS

Our idea was to equip mobile devices schema-controlled office-document editor. Our mobile site editor prototypes' measurement results concluded that there is no possibility providing the same schema-controlled editor for each mobile platform unless we restrict its operation set and style capabilities according to the most primitive mobile device environment. As a consequence we should provide an editor which can adapt to all-time document complexity and mobile device resources. Schema-controlled edition of documents needs to perform ODF schema transformations along with schema-conform document transformations. There is a need for a user-friendly formal document-frame and document-style base set from which users can freely build their desired document structure and desired style, which can be translated into a sequence of schema transformation primitives.

ACKNOWLEDGEMENT

Supported by National Innovation Office under TECH_08-A2/2-2008-0089.

REFERENCES

[1] BARNA, I. – BAUER, P.– BERNÁD, K. – HERNÁTH, Zs. – HORVÁTH, Z. – KŐSZEGI, B. – KOVÁCS, G. – KOZSIK, T. – LENGYEL, Zs. – ROTH, R. – SIKE, S. – TAKÁCS, G.: ODF Mobile Edition – Towards the development of a mobile office software, to appear in Proceedings of ICAI 2010 – 8th International Conference on Applied Informatics.

[2] OASIS Open Document Format for Office Applications (OpenDocument) TC – Open Document Format Specification, 2006–2010. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

[3] ISO/IEC 26300:2006 Information technology – Open Document Format for Office Applications (OpenDocument) v1.0, 2006. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485

[4] OASIS RELAX NG Committee Specification, December 3, 2001. http://www.relaxng.org/spec-20011203.html

[5] ISO/IEC 19757-2:2003 Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG. http://www.iso.org/iso/catalogue_detail.htm?csnumber=37605

[6] ISO/IEC 19757-2:2008 Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG. http://www.iso.org/iso/catalogue_detail.htm?csnumber=52348

[7] Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, November 26, 2008. http://www.w3.org/TR/xml/

[8] MONCINELLI, F. – INVERARDI, P.: A Resource Model for Adaptable Applications, in: Proceedings of the 2006 international workshop on self-adaptation and self-managing systems, Section: Models, ACM Press, New York, 2006, pp. 9–15.

Received June 28, 2011, accepted September 14, 2011

BIOGRAPHIES

Zoltán Horváth received his MSc in mathematics, physics and computer science in 1986 at Etvs Lornd University (Budapest, Hungary). He received his PhD (title: "A Relational Model of Parallel Programs") in 1996 and completed his habilitation process (title: "Verification of Distributed Functional Programs") in 2004 at the same university. He is head of Department of Programming Languages and Compilers since 2003, and full professor since 2008. Between 2007 and 2010 he was vice-dean for scientific affairs and international relations of Faculty of Informatics. Since 2010 he is vice-rector for international relations. He is the leader of the Budapest Associate Partner of EIT ICT Labs.

Zsolt Hernáth received his MSc degree in mathematics in 1973 from Etvs Lornd University, Faculty of Natural Science (Budapest, Hungary). He received his PhD degree (title: "On the Role of Data in Modelling the Real World: Data and what are behind them") in 2008 at the Faculty of Informatics of the same university. He was involved in a number of research and development enterprises, several industrial and business projects in Hungary and abroad. From 1980 till 1996 he worked for Etvs Lornd University Computer Centre. In 1990 he was invited by University Paderborn to take part in JESSI (Joint European Sub-micron Silicon Initiative) Common Framework (1990-1995) as guest researcher and developer. Since 2003 he has been working at the Department Information Systems of Faculty of Informatics of Etvs Lornd University, since 2008 as a senior research fellow.

Zsolt Lengyel received his MSc in computer science in 2010 at Etvs Lornd University (Budapest, Hungary). He entered the postgraduate school of the same university, where he is a PhD student.

Sándor Sike received his MSc in computer science in 1988 at Etvs Lornd University (Budapest, Hungary). He received his PhD (title: "Computer-Based Constrained Chemical Structure Generation in Three Dimensions" in 1992 at the University of Leeds (United Kingdom). He started working at Etvs Lornd University in 1988, and after finishing his PhD at Leeds he returned. He is an associate professor at Etvs Lornd University since 1996.