Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 2. pp. 313–321.

ODF Mobile Edition – Towards the Development of a Mobile Office Software*

Imre Barna[a], Péter Bauer[a], Kinga Bernád[a], Zsolt Hernáth[b], Zoltán Horváth[a], Balázs Kőszegi[a], Gergely Kovács[a], Tamás Kozsik[a], Zsolt Lengyel[c], Róbert Roth[a], Sándor Sike[c], Gábor Takács[a]

[a]Dept. of Programming Languages and Compilers
e-mail:(bib|bauer_p|bekraai|hz|pma|sanyisd|kto|rorraai|baller)@inf.elte.hu

[b]Dept. of Information Systems
e-mail:hernath@inf.elte.hu

[c]Dept. of Software Technology and Methodology
Faculty of Informatics, Eötvös Loránd University
e-mail:(lengyel|sike)@inf.elte.hu

Abstract

Open Document Format (ODF) [1, 2] is an increasingly popular office document format which is accepted by most of modern office suites and is the native format of OpenOffice.org. The aim of our project is to create a software suite of specialized mobile ODF schemata and editors to provide support for editing ODF-based documents on mobile devices, such as mobile phones, smartphones and PDA's. Currently editing of concurrent document formats is supported on smartphones and PDA's only. We created prototype software to test editing capabilities of a wide range of mobile devices, tested them with a variety of schema versions of different complexities and measured the resource need of different editor operations through a scripting interface. This paper presents our prototype tools, test methodology and results.

Keywords: Mobile office, Mobile software, ODF, XML

MSC: 68U15 Text processing, 68U35 Information systems

1. Introduction

The spread of Open Document Format as a desktop document format implies that – similarly to concurrent formats, like those Microsoft Office have – ODF supporting

*Supported by NKTH under TECH_08-A2/2-2008-0089.

editor suite running on mobile devices is needed. The goal of our research project is to develop a data model, a schema driven editor model implemented on mobile devices and a supporting desktop tool set. Considering the hardware and software resources available on mobile platforms, we design editor software which can provide the widest variety of ODF features while ensuring user-friendly edition of standard conforming documents. As target devices range from cell phones to smartphones to PDA's and have different limits of different resource aspects, the software suite's resource needs should be adaptive to the current platform.

To reach our goal, we investigate the resources available to editor software, and the resource need of primitive editor operations. Based on the analysis we select ODF features that can be implemented in a user-friendly and resource-efficient way on low-resource devices. We will show that a data model different from, but based on ODF, is necessary to implement such an editor, thus we develop a desktop tool set supporting the transformation of documents, and ensuring ODF conformity of documents edited on the mobile device. As ODF standard is developing in time, it is important that edited documents always conform to the standard.

First we surveyed the capabilities of mobile devices and developed a method to define the document schemata used on mobile devices. We created test software measuring the resource need of primitive editor operations and collected data of execution time, memory usage and energy profile against complexity and size of the document and its schema as well as different editor versions supporting different set of features. We introduce our test software suite in section 2.

To address document complexity we defined a list of reduced ODF schemata based on typical users and use cases of office document editors. Another aspect is the size of the editor determined by the set of available operations and document complexity through the document schema. For schema and document transformation and distribution in the test period, we created a set of supporting tools. Transformed schemata and document set used in our tests are described in section 3.

While testing editor functionality in detail we also investigated whether ODF document layers (content, style, metadata and editor options) can be handled more efficiently when weaved into a single XML document [6] or divided into separate ones. We developed a layered document model in which structural, content and style information as well as their changes are stored in 6 separate layers. To support mobile devices with such heterogeneous resource limits, we tested a variety of schemata of different complexity and decided to have an adaptive set of schemata instead of using a common schema on all platforms. The decision enables us to follow the evolution of hardware and software architecture. Test results and conclusions are described in section 4.

In section 5 we give an overview of other office software running on mobile devices. We plan our future work in section 6.

2. Software Tools

2.1. Test Editors

To support the possibly widest range of platforms, CLDC 1.1 and MIDP 2.0 featured J2ME environment has been chosen as base requirement. This software platform is already provided by most mobile devices available today. Choosing the right environment, we took a great care of not excluding commonly used mobile devices still not accepting an unreasonably narrow feature set either (e.g. MIDP 1.0), because the latter would occasionally have led to the exclusion of required J2ME library functionality (e.g. XML file processing library functionality – base for ODF document handling).

We have designed and implemented a text editor prototype that beside primitive text editing functionality (like displaying paragraphs, inserting, deleting and marking of parts of texts) also supports elementary operations for styles (like creation of styles, applying styles, selection of font faces and font styles, colors both for characters and background). A spreadsheet prototype has also been implemented, which supports displaying and editing tables in appropriate formats, just as displaying spreadsheets, data type conform displaying and editing cells of tables.

We included a typeless XML parser in our editor prototype which is able to build a DOM-like tree of the document loaded to be able to test a variety of different mobile site ODF schema derivatives. Low-resource logging has been added to test editors that logs the starting and finishing time of certain operations. Timestamps are saved at the end of each run to avoid latency caused by I/O operations. We also log the memory usage throughout the tests. This allows us to create operation performance and memory benchmarks. In addition, timestamps also helps connecting the data collected with profiler tools (like Nokia Energy Profiler) with editor operations. We could collect CPU load, energy benchmarks and wireless transmission data this way on mobile phones running Symbian OS.

The editor contains a lightweight script language. The language consists of basic selector statements like `setcursor` and `select`, style-modifier statements like `createfontstyle` or `createrandomfontstyle`, and document modifying statements, like `applystyle`, `inserttext` or `deleteselection`. During script run, each document-modifier statements are applied on the current selection of the text. The syntax of the language is based on CSV, where the first field of each row is a statement followed by its parameters. The language supports for-loop with an integer loop-variable. Loops can be nested, and the loop-variable can be referenced inside the loop body by squared parentheses.

2.2. Desktop Tool set

Various tools has been created to ease the creation of schema derivatives and conforming documents that can be transferred to mobile devices. We developed a RELAX NG [3, 4, 5] schema editor with a GUI which can search for certain XML elements, RELAX NG patterns and save semantically valid schema derivatives after

removing selected elements. It also has a command-line interface to batch process a list of patterns to remove from the given schema. With this tool we created a list of schema derivatives to test the editor with. The tool also outputs a list of XPath expressions describing those sub-trees of the ODF documents which have to be removed for the document being conform with the new, reduced schema.

We also created a document transforming tool to reduce test documents. It processes the list of XPath expressions, reduces the documents, while also removes namespace declarations that cannot be validated on the mobile site. After the transformation we get a more compact document, which can be edited using significantly less resources. After editing the document we want to restore the removed contents, thus every unnecessary sub-tree in the document is replaced by a typed marker. Marker types enables the mobile -site editor to display icons in place of removed fragments (like big pictures, complex tables, etc.). Knowing the type of the removed fragments is also a key if we want to perform editor operations around markers as we can guarantee document validity this way. As markers have unique ID's we can put the replaced fragments back into the edited document while uploading to the desktop machine with another tool. After the transformation the document is conform to ODF schemata.

Test software communicating over 3G network and Wi-Fi has been created, to model document and metadata transmission between the mobile and desktop site. We compared the time needed to process zipped and plain-text XML files containing spreadsheets. Our tests have shown that the overhead of compressing and decompressing files is very low, and the smaller file size does not only speed up the transmission but also decreases load times from the flash memory installed in mobile devices. As loading data proved to be a bottleneck in the tests, we decided to use zipped files in all of our later tests.

3. Test Schemata and Document Set

Starting from the ODF Standard, we investigated the features offered and controlled by the ODF schemata, such as tracking changes, or accessing data sources. We have defined different concept categories of the standard, and several user classes the different categories are assigned to. By gathering XML tags describing different categories, different reduced schemata, each tailored to the corresponding user class, could be achieved. These simplified schema derivatives constituted the base of test environments of our benchmarks. According to our approach of layered structure, to measure the resource need of all user group established, all simplified schema derivatives have been generated with and without embedded style information. User classes and corresponding concept categories are shown in table 1.

Concepts and their ordering are subjective. They have empirically been created to simplify our benchmarks. The simplest schema we start to test with contained neither of the above concepts. From step to step we augmented them by XML tags supporting layout, enumeration, etc., and examined the text editor controlled by the step by step flaring schema. Documents of different complexity have been

created in several sizes. Document generator has been implemented to create documents of different size by multiplying the content of shorter documents. The generator guarantees unique ID's of affected document elements.

Layout Editors	Paragraphs Breakpoints
Enumerators	Enumerations
Spreadsheet Users	Spreadsheets
Multimedia Users	Shapes Pictures, Motion Pictures Objects, Formulas
Note Makers	Notes, Footnotes Headings, Foots
Publishers	Bookmarks References Bibliography
Team Users	Change Tracking
Automating Users	Macros
On-line Users	Data Sources
Dynamic Document Users	without any restriction

Table 1: Concept categories

4. Test Results and Conclusions

To develop test software we investigated a variety of mobile device emulator software. We found that the emulator included in Sun NetBeans provides the most functionality and its interface is very close to real life devices. Unfortunately due to the loose security settings it has, programs tested with the emulator were not guaranteed to run on mobile phones. Because of the processor architecture of desktop computers is different from that most of the mobile phones have, the performance tests ran in emulator environment gave results significantly different from mobile phones. The most decisive differences found and to be handled are as documented next.

4.1. Significant Settings and Operations on Mobile Devices

4.1.1. Security Settings

Examined mobile devices showed different behavior concerning security settings. Executing particular I/O operations, a number of devices pop up messages about security concerns users have to confirm in order to gain access permission to resources an operation in question addressed by. Others allowed us to give permis-

sions on the first run, or each time before starting the editor. but others gave pop-up messages before every disk access. Mobile benchmark tests modified to ensure user reaction is not measured.

4.1.2. Power Saving Mode

A commonly known behavior of mobile devices is a kind of power saving mode functioning: if users do not touch the screen or do not press a button for some minutes, the back-light reduces or the screen turns off. Our measurements showed that on some devices when the screen is turned off, processor clock rate is also reduced, which caused that running some scripts that could be completed in 10 seconds otherwise took thousands of seconds.

4.1.3. Energy Optimization

Energy optimizations takes place even when devices run in user-interactive mode. This caused that some devices executed simpler operations slower than more complex ones, as the latter caused higher processor-utilization which triggered a higher clock rate. As clock rate is adjusted with a threshold, executing a loop of simple and then a loop of complex operations took more time than executing the loops in the opposite order.

4.1.4. Spreadsheet Settings

Spreadsheets coming from different sources built by different ODF implementation in different desktop ODF software caused divergent results. OpenOffice.org does not store empty cells in spreadsheets individually, but uses ranges instead. Most other office suite stores every empty cell that resides between non-empty ones. Differently stored empty cells do not cause differences in file sizes since ODF consists of zipped XML files, but rather significant parse time after decompression.

4.1.5. Text Insertion

Our measurements showed that insert operations that needs re-fragmentation are very costly considering processor time. The cost is directly proportional to document size, but re-fragmentation can be limited by screen and paragraph size. Re-fragmentation of documents are structural operations. As a consequence of that it is reasonable to distinguish structural and textual changes. Structural and textual changes are to be separated and treated as operations on document structure and document content. To make structural operations efficient, not only the style information including automatic styles are to be detached from the content layer, but rather a separated structural layer also has to be established. The associated content can be edited through a sliding window, and therefore re-fragmentation has to be computed and performed for only the sliding window.

4.1.6. Parsing Documents

Benchmark results showed that processor time needed to parse documents increases proportionally to both document size and complexity. Since editing documents of less reduced or full complexity exceed an acceptable time on some mobile devices. With an eye on the above, an adaptive assignment of schemata of different complexity to mobile devices equipped with different resources is reasonable. We have defined different versions (cf. user classes) adaptively to mobile device resources. The formal descriptions of those are RELAX NG schemata and sets of XPath expressions. Adaptively reduced ODF schemata are defined by the corresponding software.

4.1.7. Document Size

Some mobile devices cannot open longer documents (50-100 pages), while others can not perform operations on them. Medium-sized documents (10-15 pages) on particular mobile devices have reached a J2ME run-time memory limit of 2 MB leading to an exception, or garbage collection caused an enormous performance loss. According to our measurements a high category device like Nokia E71 can even handle fulfilled spreadsheets of 400 times 400 cells, whilst an ordinary mobile phone like Sony Ericsson w890i is able to handle only 100 times 100 cells.

4.2. Layered and Segmented Documents

Test runs showed that performing textual and style modifications separately needs less resources than performing them simultaneously. It has also been discovered that structural information should be distinguished. Therefore we propose a layered model of 6 XML documents: 1. Structural information, 2. Style information, 3. Textual information with references to positions in document structure and with references to styles applied on text portions, 4. Changes in document structure, 5. Changes in styles and 6. Textual changes including changes to references. When we start to edit a document the first three layers need to be downloaded from the desktop. After editing the document, the latter three need to be uploaded. As we upload the change-lists of the three layers, conflict resolution can be implemented if distributed editing is in need.

As processing the structure of a whole document is resource-intensive and the sliding window used for editing the document is much smaller than that of a desktop editor, it is reasonable to segment documents and process only few segments on a mobile device at a time. This way the whole document do not need to reside in the memory while editing, but we can download selected segments from the desktop on demand. This solution creates an opportunity to edit the same document concurrently on multiple devices as locks or conflict resolution has to be applied only for a limited number of edited segments.

5. Related Work

Before starting our development we investigated mobile office suites supporting ODF. mOOo project started by Java.net group (https://mooo.dev.java.net/) aimed at handling ODF text documents and spreadsheets on mobile devices, however they have not started their development yet. In the last two years they released a remote control application for desktop presentations, called Impress Controller. Sept Solutions developed Mobile Office running only on Symbian OS (http://www.sept-solutions.de/English/office.php). It can display ODF documents but cannot edit them. RedOffice suite offers edit features (http://www.redoffice.com/?class=sy), but is running only on MID and UMPC devices, not on handheld devices. Visor ODF Movil is a Spanish J2ME based software running on smartphones and PDA's (http://visorodfmovil.morfeo-project.org/). It can display ODF documents, but cannot edit them. Other office suites like QuickOffice, supports only Microsoft Office formats, not ODF (http://www.quickoffice.com/). We concluded that an office suite running on a wide range of mobile devices and capable of editing ODF documents is not present on the market yet.

6. Future Work

Working with structural elements, hierarchical limits can be defined. If the mobile device has lower capacity, the recursion depth of recursively embedded elements (like enumerations or tables) can be limited. This type of reduction leads to a document which is a sub-document of the original, while the schema allowing only a limited depth of recursion is not strictly a sub-schema of the original as it includes new non-terminal patterns. We are working on a schema transformation descriptive language to be able to express these kinds of reductions.

Our measurements have revealed two problems. The loading time of the document and the memory needed for document representation can be critical. Both can be reduced by an editor which is built on a memory- efficient representation of the schema, which can be processed efficiently. We are working on such a data model to build a schema driven editor software. We are also developing a resource model to support resource-adaptivity of the editor [7].

References

[1] 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

[2] 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

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

[4] 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

[5] 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

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

[7] Mancinelli, 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, Pages 9–15. ACM Press, New York (2006)

Faculty of Informatics, Eötvös Loránd University
Pázmány Péter sétány 1/C H-1117 Budapest, Hungary