電子書籍 XMDF のマルチプラットフォーム技術

広沢昌司† 中村雅也‡ 菅野充臣‡ 中井宏樹† 沢田裕司‡ 田中秀明‡
シャープ株式会社 研究開発本部 ソフトウェア開発センター†
シャープ株式会社 研究開発本部 プラットフォーム開発センター‡

1. はじめに

近年、電子書籍が本格的に広まってきており、多くの種類の端末で電子書籍を閲覧することが求められているが、端末の能力や開発・動作環境が様々な中で品質の高いアプリケーションを多くのプラットフォーム向けに効率よく開発するには優れた技術と開発手法が求められる。

弊社の電子書籍技術 XMDF の開発グループでは、低い処理能力しか持たない端末でも高速に閲覧可能な電子書籍技術を開発し、また二十を超えるプラットフォームでのビューア開発を高品質・低コストで行ってきた。

本稿では、これらの開発を可能にした電子書籍技術と開発手法について述べる。

2. XMDF の概要

XMDF(ever-eXtending Mobile Document Format)は、弊社が 2000 年から開発してきた電子書籍技術である[1]。小説などのテキスト系コンテンツ、辞書検索が可能な辞書系コンテンツ、コミック系コンテンツまで多くの形式をサポートし、暗号化や改ざんチェックなどの DRM 機能も備えている。2009 年には XMDF をベースとした 2 つの IEC 国際標準規格[2]が発行された。

XMDF ビューアはμITRON, Palm, Zaurus, MS-Windows CE, Symbian, BREW, Linux, MS-Windows など多くのプラットフォーム向けに開発され、数百機種に搭載されている。

3. マルチプラットフォームでの開発手法

ソフトウェアを高品質に開発する開発手法は数多く存在するが、一つのソフトウェアを多くのプラットフォームに向けて効率的に開発するには、一般的な開発手法に加えていくつかの開発手法／設計／方針をとることが重要である。

Multi Platform Technologies of Electronic Book "XMDF"
† Software Technology Development Center, Corporate R&D Group, SHARP Corporation
‡ Platform Technology Development Center, Corporate R&D Group, SHARP Corporation

3.1 低い処理能力の端末を意識した設計

様々なプラットフォーム／端末に対応するには当初から低い処理能力の端末を意識した設計が重要となる。

XMDF のテキスト系コンテンツは、PDF などのように文字のサイズや位置が固定された静的なレイアウトではなく、HTML などと同様に文字のサイズを変更可能な動的なレイアウトを採用している。

一般的な HTML ブラウザでは、HTML ファイルを全てレイアウトして表示するため、レイアウト全体を作成するのに処理時間がかかり、必要とされるヒープ量が大きくなる問題がある。

XMDF ではこの問題を解決するため、図２に示すようにテキストデータである XML ファイルを分割し、表示に必要な部分の分割テキストデータだけを読み込んでレイアウトするようにしている。

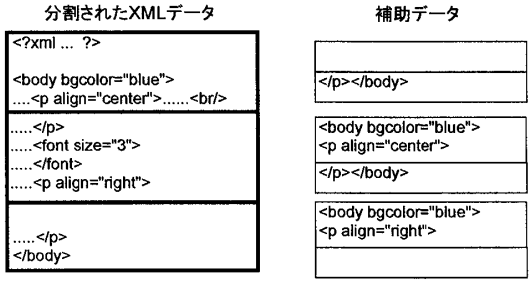

図１　XML データと補助データの例

しかし、単に XML ファイルを分割するだけでは当然、valid な XML ファイルとはならず、また、レイアウトの修飾情報（図 1 の p タグなど）も失われてしまう。

そこで分割された XML ファイルの前後のタグを取りだし、分割された XML ファイルの前後にそのタグの文字列を結合して解釈することで、valid な XML として処理することができる。

上記の仕組みにより、低い処理能力の端末であっても、高速かつ省メモリのレイアウト処理を行うことが可能になる。

3.2 branch を trunk に戻すソース管理方針

ソフトウェア開発にソースコード管理システムは必須であるが、共通ソースを使って複数の製品を並行開発する場合、開発の終盤になると、他製品向けの改修による副作用をさけるため、製品単位で branch に移行することが一般的である。後継機種の開発で大きな機能拡張を行わない場合は、既存 branch からさらに新しい branch を作成するというようなこともしばしば行われる。

しかし、このように branch を広げていくと製品の数だけ branch ができてしまい、メンテナンス作業が膨大になってしまう問題が発生する。

これらの問題を未然に防ぐため、以下の方針を採用した。

（方針１）branch への改修は trunk にも常に反映させる。

（方針２）後継機種の branch 化をタイミングを見て止め、trunk ソースに戻す。

3.3 自動ビルド、自動テストの導入

ある機種向けでは問題ない改修が他の機種では不具合を発生させてしまうことがマルチプラットフォーム開発では起こりやすい。具体的にはビルドが通らなくなる不具合と、ビルドはできるが動作がおかしくなってしまう不具合に分類できる。

これらの不具合を防ぐため、自動ビルドと自動テストのシステムを構築した。

自動ビルドは、定期的かつ自動的に各機種向けのソースを取得・ビルドし、結果をメールでレポートするシステムである。

自動テストは、自動ビルドで生成されたプログラムを使って自動的に単体テストや回帰テスト、ストレステストを行い、結果をメールでレポートするシステムである。自動テストは、ユーザー操作やコンテンツ読み込みなどのテスト手順をファイルで記述し、順次解釈・実行することで行っている。XMDF ビューアは表示結果が正しいことが重視されるため、描画ログをファイル出力し、正解ログと比較することで不具合が発生していないかチェックしている。

自動テストシステムは共通エンジン部分で実装しているため、実機上で実行することもできる。

マルチプラットフォーム開発での本システム導入の主なメリットは以下である。

（メリット１）各機種での不具合改修を正確かつ効率よく行えるようになった。

不具合の正しい改修方法はオリジナルの改修を行った者が一番詳しいが、実際の開発現場では社内外の開発メンバーの入れ替わりが起こる。そのためある機種での改修から時間が経った後に他機種での不具合が発覚しても詳しくない者が対応せざるをえない場合がある。そのため正しい改修方法を探るのに時間を要する問題があったが、このシステムの導入により解消された。

システム導入以前は、新規機種・新規後継機種の開発を trunk ソースから行う時にビルドや動作の不具合の改修に長い時で１週間ほどかかっていたものが、現在ではほぼゼロとなった。

（メリット２）branch ソースから trunk ソースへの切り替え(3.2 の方針２参照)リスクが減った。

これは、回帰テストなどにより動作が変わっていないことが保障できるためである。

3.4 コンフィギュレーションツールの導入

各機種での機能拡張／カスタマイズ要望に対応し、プラットフォーム間の機能の違いなどを吸収させるため、機能のコンフィギュレーション項目が増えてしまうことがある。

項目が増えると設定漏れが起きるなどの問題が発生しやすくなるため、全ての機種のコンフィギュレーションを表形式で管理し、各機種用のヘッダファイルなどを自動生成するツールを開発した。

表形式にすることで、設定漏れを防ぎ、後継機種での設定作業が容易になった。

4. まとめ

マルチプラットフォームを意識したソフトウェア開発では、本論文で示したような長期間の継続開発を見越した設計や開発手法／方針の採用が重要である。今後も対応するプラットフォームは増え続けていくと思われるので、さらに低コストで高品質なソフトウェア開発を実現する手法を探っていく必要がある。

5. 参考文献

[1] 北村　義弘, 岩崎　圭介, 田中　秀明：”電子書籍と XMDF 技術”, シャープ技報.No.84, pp13-17(2002)

[2] IEC 62448 Edition 2.0 : http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/42651,
IEC 62524 : http://webstore.iec.ch/Webstore/webstore.nsf/Artnum_PK/42619