社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ＦＩ－86 (12)
2007－ＤＤ－60 (12)
2007／3／27

組込み機器上でWebサービスを扱うプラットフォームの構築

Improving User Experiences with Services for Embedded Devices

中西正洋†＊ 坂倉健太郎＊ 財満博昭＊ 畑山尚毅＊ 小野修一郎＊ 尾上孝雄†
Masahiro Nakanishi Hiroaki Zaima Kentaro Sakakura Syuichiro Ono Takao Onoye

†大阪大学 情報科学研究科
＊シャープ株式会社 技術本部 プラットフォーム開発センター

概要

近年，PC の世界では Web サービスを利用して，新しい形のユーザエクスペリエンス（user experience）を体験できるようになってきている。一方，携帯電話や薄型テレビのような組込み機器の世界でも，各機器がインターネットに接続されつつある。しかしながら，組込み機器での Web サービスの利用は始まったばかりであり，PC と同じようにユーザは必要に応じて Web ブラウザを立ち上げ，所望のサービスに接続している。そのため，組込み機器に適した Web サービスの利用法を確立することが課題となっている。そこで，ユーザが能動的に Web サービスを利用するのではなく，ユーザが必要としている Web サービスを組み合わせて，自動的に UI に表示できるプラットフォームを構築することを，本研究の目的とする。

By using PC, end-users are now able to enjoy variety of new user experience through web services. Embedded devices such as mobile phones and flat-screen TVs are also enjoying network connectivity but with the condition that end-users have to launch a web browser when using web services. This paper illustrates a method to let end-users directly use web services without launching a web browser on the embedded device.

1．はじめに

近年，PC の世界では Web サービスを利用して，新しい形のユーザエクスペリエンス（user experience）を体験できるようになってきている。例えば，ブログやソーシャルネットワーキングサービス（SNS）により，ユーザ参加型の新しいコミュニケーションを体験ことが可能となった。また，多数のユーザが部分情報を Web に持ち寄る集合知を利用して，Web 辞典やソーシャルブックマークサービスを利用できるようになった。このような現象はティム・オライリー（Tim O'reilly）により Web2.0[1]と名づけられ，日常生活においても使われるようになってきている。

一方，携帯電話や薄型テレビのような組込み機器の世界でも，各機器がインターネットに接続されつつある。その結果，Web ブラウザや Java プラットフォーム上のアプリケーションソフトウェアを利用して，PC 以外の機器で Web サービスを利用できるようになってきた。

しかしながら，組込み機器での Web サービスの利用は始まったばかりであり，PC と同じようにユーザは必要に応じて Web ブラウザを立ち上げ，所望のサービスに接続している。組込み機器は，機器ごとに用途が異なり，保有する表示デバイスや入力デバイスに大きな差がある。そのため，組込み機器に適した Web サービスの利用法を確立することが課題となっている。そこで，ユーザ操作によって Web ブラウザを立ち上げ，能動的に Web サービスを利用するのではなく，ユーザ必要な Web サービスを組み合わせて，自動的に UI として生成できる組込み機器向けのプラットフォームを構築することを，本研究の目的とする。

なお，組込み機器の UI の開発では，開発コストを削減のため，UI を XML に代表されるマークアップ言語で記述し，システムソフトウェア本体から分離する流れが広まってきている。その結果，従来の組込み開発者だけでなく，デザイナーが組込み機器の UI の開発を担えるようになり，開発コストが削減されつつある。このような流れを受けて，本研究では，組込み機器のプラットフォームを構築するにあたり，UI をマークアップ言語で記述するアプローチを取る。

以下，2 章で既存技術の紹介をし，3 章では，マークアップ言語で記述された UI を利用しつつ，Web サービスを利用できるプラットフォームの構築を行う。最後に，4 章で今後の展望とまとめについて述べる。

2．既存技術

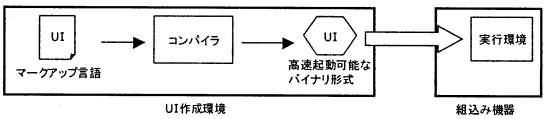
2．1．UI をマークアップ言語で記述する既存プラットフォーム

マークアップ言語を用いて UI を記述可能なプラットフォームとして，以下の 2 つのタイプが存在する[2)]。

（1）UI 作成時に高速実行できるバイナリ形式に変換するプラットフォーム

このタイプには，アクロディア[注1)]の VIVID UI[注1)]，Microsoft[注1)] 社の Automotive UI Toolkit[注1)]，QUALCOMM[注1)] 社の uiOne[注1)]，UIEvolution[注1)] 社の UIEngine[注1)] が存在する。UI 作成時にバイナリ形式に変換するメリットとして，UI の起動速度が速い。逆にデメリットとして，UI 開発のための専用ツールが必要となる。また，組込み機器上で UI の一部を交換しようとした場合，高速起動可能なバイナリ状態になっているので，他のアプリケーションからの流用が困難であるというデメリットも存在する。

Fig. 1 UI 作成時にバイナリに変換



注1) ここでの掲載の各社名および各製品名は，商標または登録商標である。

（2）マークアップ言語のまま組込み機器上で実行するプラットフォーム

ACCESS[注2)] の NetFront Dynamic Menu[注2)] がこのタイプである。このタイプのプラットフォームのメリットとして，HTML などの標準仕様を用いて UI を作成するため，PC 用の UI 開発ツールが流用でき，PC 上でプレビューが容易である。また，組込み機器内でも標準仕様のマークアップ言語のまま保持しているため，他のアプリケーションからの利用が容易であるというメリットが存在する。デメリットとして，組込み機器上で XML パースを行うため，UI の起動速度が遅いという問題点がある。

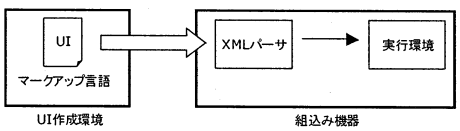


Fig. 2 マークアップ言語の状態で起動

注2) ここでの掲載の社名および製品名は，商標または登録商標である。

2．2．UI を記述するマークアップ言語

本研究では UI を記述する言語として，ベクターグラフィック用の XML 言語である SVG 言語と，スクリプト言語として ECMA スクリプトを利用している。

以下に，説明を行う。

（1）XML

XML[3]は，データを記述するマークアップ言語を定義するためのメタ言語である。W3C（World Wide Web Consortium）により1998年2月にXML 1.0が勧告された。2007年現在，W3C勧告の最新バージョンはXML 1.1である。

（2）SVG

SVG[4]は，XMLによって記述されたベクターグラフィック言語である。W3Cで2003年1月にSVG1.1がW3C勧告となっている。現在SVG1.2が策定中である。簡単な矩形を描画するSVG記述の例を以下に示す。

```
<svg>
  <rect x="10" y="20" width="80" height="70" style="fill:red; stroke:red;" />
</svg>
```

Fig. 3 SVGによる矩形の記述例

（3）ECMAスクリプト

標準団体であるECMA[5]が「ECMA-262 ECMA Script Language Specification」として標準化しているスクリプト言語である。

3．プラットフォームの構築

本研究では，ユーザが能動的にWebブラウザを立ち上げ，所望のWebサービスを利用するのではなく，ユーザの周囲の状況に応じて必要なWebサービスを組み合わせて，自動的にUIに表示できる組込み機器上のプラットフォームを構築する。

本プラットフォームを構築するために以下の手順で進めていく。本稿では（3）までを行う。

（1）UIの記述法の定義

（2）UIの高速起動可能なアーキテクチュアの設計

（3）Webサービスを利用するアーキテクチュアの設計

（4）ユーザの周囲の条件に応じて，ユーザが必要なWebサービスを検出する方法

3．1．UIの記述法の定義

本プラットフォームでは，開発コストを削減しつつ，組込み機器上で動的にUIを生成するため，UI記述を4つの部分に分割する。これらをUI記述のUIの部分記述と呼ぶ。

・データ記述

・レイアウト記述

・ルール記述

・スクリプト記述

順に説明を行う。

3．1．1．データ記述

データ記述では，データidとそれに対応する文字列や画像のリソースのファイル名を記述する。具体的には，参照する画像フォーマットとして，PNG画像やJPEG画像を用いている。文字列として，アイコン名称や起動するアプリケーション等が設定されている。独自のXML言語でリソースは記述する。

```
<xml>
  <image id="icon01" xlink:herf="icon01.png" />
  ・・・[データidと対応する画像リソース]
  <image id="icon12" xlink:herf="icon12.png" />
  <text id="icon_name01">インターネット</text>
  ・・・[データidと対応する文字リソース]
  <text id="icon_name12">設定</text>
  <text id="ap01">../internet.ap</text>
  ・・・[データidと対応するアプリケーション]
  <text id="ap12">./setting.ap</text>
</xml>
```

Fig. 4 データ記述の例

3．1．2．レイアウト記述

レイアウト記述では，データ記述の各リソースが配置される可能性のある位置を定義する。レイアウトidと座標で構成される。SVG言語で記述する。

3．1．3．ルール記述

ルール記述では，データidをレイアウトidにマッピングするルールを記述する。対応関係をメモリ

上に保持している。

3．1．4．スクリプト記述

スクリプト記述では，各レイアウト id に対して，キー入力が発生したときのアクションを記述する。代表的なアクションとして，アプリケーションの起動と，別のレイアウト id にフォーカスが遷移することの2タイプがある。ECMA スクリプトで記述する。

```
<svg>
  <image id="pos01" x="0" y="0" />
  ・・・[レイアウトidと位置]
  <image id="pos12" x="160" y="240" />
</svg>
```

Fig. 5 レイアウト記述の例

```
pos01→icon01
...
pos12→icon12
```

Fig. 6 ルール記述の例

```
<script>
  if (key=[右キー]) {
    [フォーカスを右に移動させる処理の実装]
  }
  ...
  if (key=[決定キー]) {
    [アプリケーションを起動させる処理の実装]
  }
</script>
```

Fig. 7 スクリプト記述の例

このような UI 記述法を採用することにより，組込み機器上で各記述を組み合わせて UI を生成できるようになる。

つまり，レイアウト記述を変更すれば，UI のレイアウトだけを変更可能となり，スクリプト部分を変更すれば，キー操作に対する UI の振舞いだけを変更になる。また，データ記述だけを変更することで，UI の見た目を簡単にカスタマイズすることが可能となる。さらに，ルール記述だけを変更すれば，メニュー項目の入れ替えが簡単に交換可能となる。

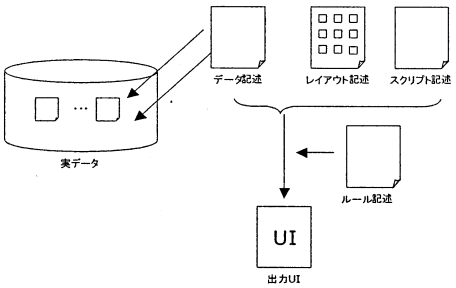


Fig. 8 UI の部分記述から UI の作成

3．2．起動の高速化に関して

マークアップ言語を用いて UI を記述する組込み機器上のプラットフォームとして，以下の2つのタイプがあることを説明した。

（1）UI 作成時に高速実行できるバイナリ形式に変換するプラットフォーム

（2）マークアップ言語のまま組込み機器上で実行するプラットフォーム

マークアップ言語で UI を記述することにより，組込み機器内で XML を扱う各種処理ルーチンを使える。そのため，組込み機器で動的に UI を作成するためには，利便性を考慮して，組込み機器内においても，マークアップ言語の状態で保持する方が望ましい。しかし，その場合，UI を起動するたびに，パース時間がかかる問題が発生する。

Table. 1 比較表

	(1)	(2)	提案システム
UI 起動速度	高速	低速	高速
マークアップ言語のまま起動	不可能	可能	可能

そこで，本研究では，UI をマークアップ言語のまま組込み機器内に保持しつつ，UI の高速起動が可能

となるプラットフォームのアーキテクチュアを提案する。

すなわち，UI の部分記述のうち，データ記述と，レイアウト記述と，スクリプト記述だけを組込み機器上で高速起動可能な状態に変換し，UI 起動時にルール記述に従って，高速起動可能な 3 種類の状態を組み合わせて UI を生成し，実行する。この際，レイアウト記述とスクリプト記述はセットにして変換を行い一つの高速起動可能なバイナリ形式であるデータを生成する。

高速に起動できる状態に UI 記述を展開すると，使用メモリサイズが増えるが，ユーザが利用する UI だけを高速起動できる状態に変換するため，（1）の場合と比較して，使用メモリサイズを節約できるというメリットも生まれる。

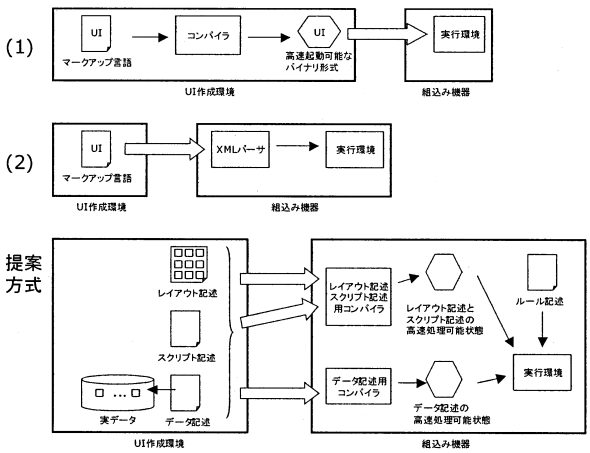


Fig. 9 プラットフォームの全体構成の比較

3．3．Web サービスの利用に関して

一般にネットワークアクセスを行うと，消費電力量が増え，消費電力量を下げることが課題となる。そこで，組込み機器内にサービスプロキシをおき，これを介して通信を行い，Web から情報を取得することで，ネットワークアクセスの頻度を下げるコントロールを行い，問題の解決を行う。

サービスプロキシの別の効果として，ネットワークに一時的に遮断されている場合でも，プロキシのデータを表示することができるので，UI に Web サービスの情報を表示することが可能となっている。

但し，デメリットとして，最新の Web のデータが UI に反映されないという問題がある。リアルタイムに情報を取得必要な，例えば組込み機器の GPS データと連動した地図上の位置を表示する Web サービスなどでは，プロキシを無効にして利用し，1 日数回しか更新されないニュースサイトによって配信される RSS データの取得にはプロキシを有効にして利用することで回避可能である。

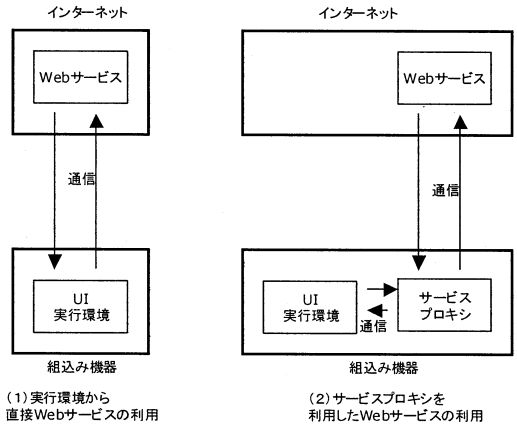


Fig. 10 Web サービスにアクセスする仕組み

4．今後の展開とまとめ

本稿では，Web サービスを利用可能な UI を生成する，Web ブラウザでない組込み機器上のプラットフォームを構築した。

まず，マークアップ言語で UI を記述する方法を述べ，次に UI の高速起動を行うために，必要に応じて UI の部分記述を高速起動できる状態に変換する方法について述べた。最後に，サービスプロキシを介して Web サービスを利用する方法について述べた。

今後は，UI 起動時間に関して，UI 記述の一部を組込み機器で高速化することによる改善効果と，サー

ビスプロキシの有無による UI 起動時間の改善効果に関して，エミュレータ上で計測し，定量的に評価していく計画である。

さらに，組込み機器内の様々な情報にメタ情報[6]を付加し，その情報と他のメタ情報を元に，場所や時間等の条件から判断して，UI を生成する研究を行う予定である。その結果，ユーザが組込み機器のディスプレイを覗き込むだけで，周囲の組込み機器と連携[7][8]し，いつでもどこでも必要な情報が UI に表示されている世界を実現していく。

参考文献

1)O'REOLLY Web Site, "What Is Web 2.0," (オンライン), <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>, (参照 2007.2)

2) 竹居智久, "組み込みソフトをXMLが変える 開発効率向上の切り札に," 日経エレクトロニクス, pp.51-58(2005,7).

3)Extensible Markup Language (XML), (オンライン), <http://www.w3.org/XML/>, (参照 2007,2)

4)Scalable Vector Graphics (SVG), (オンライン), <http://www.w3.org/Graphics/SVG/>, (参照 2007,2)

5)European Computer Manufaclurers Association, オンライン), <http://www.ecma-international.org/>, (参照 2006,10)

6)神崎正英, "セマンティック・ウェブのための RDF/OWL 入門," 森北出版株式会社(2005).

7)井垣, 中村, 玉田, 松本, "サービス指向アーキテクチュアを用いたネットワーク家電連携サービスの開発," 情報処理学会論文誌, 46, 2, pp.314-326(2005).

8)中西ほか, "Webサービスと組込み機器の機能をシームレスに扱うプラットフォームの構築," シャープ技報, 95, pp.58-62(2007).