

組み込み機器向けブラウザの開発

佐々木幹郎* 山中弘* 齋藤正史*

組み込み機器向けのブラウザ・メーラソフトウェアについて再利用性に着目した開発を行った。まず、携帯電話向けブラウザ・メーラ開発において、様々な機種で共通に使用されるコンテンツデータ処理モジュールに着目し、流用可能なソフトウェア・コンポーネントとしての設計・実装を試みた。また、実際に開発モジュールをカーナビゲーションシステム向けブラウザ開発に流用し評価を行った。基本的なデータ構造および処理構造を変えることなく流用が可能であり、コンポーネントとしての有効性を実証できた。また、更なるモジュールの分割・統合を行うと同時に、拡張性を備えつつ省リソースを実現するためにプラグイン機構を備えたブラウザの再設計を実施した。

Development of Web Browser software for Embedded Systems

Mikio Sasaki Hiroshi Yamanaka Masashi Saito

We developed embedded browser/mailer software that focused re-usability of software component. First, in mobile phone software development, we tried to design and implement contents data processing modules as software components which can adapt to various series of mobile phone. Then we actually apply those components in ca navigation system software development, and evaluate its reusability of component and processes. Those components are available to reuse for content analyzing, because it need not to change data structure and logic when the component adapted to other systems. And we aim to high expandability in low resource by designing and implementing module interfaces according to plug-in system.

1.はじめに

組み込み機器の分野では、短い開発期間で不具合のない高信頼のソフトウェア開発が求められている。さらに、競争の激化から組み込みソフトウェアは年々高機能化・複雑化してきており、開発効率の向上が急務である。

開発効率向上には、ソフトウェアの部品化・流用は有効な手段の一つである。しかしながら、組み込み機器の分野では、機種毎に多様に要求仕様が変化すること、厳しいハードウェアリソース制約を満たしつつ十分な性能を実現しなければならないことなどから、対象機種独特の設計・実装が行われ、別の機種への流用が難しくなる場合が多い。

我々は、開発規模が大きく様々な機種に展開可能な処理モジュールとして、まず携帯電話向け

のブラウザ・メーラソフトウェアの開発において、要求仕様・ハードウェアの相違に柔軟に対処可能とする部品の開発を行った。さらに、実際にそのモジュールをカーナビゲーションシステム(以下「カーナビ」と記述)向けブラウザ開発に流用・評価した。また、それらの開発を通して得られた課題に基づき、再利用性の向上と限定されたリソース内で機能的なスケーラビリティを確保するためのプラグイン機構を備えたブラウザの設計を試みたので報告する。

2.携帯電話向け組み込みブラウザ・メーラの開発

2.1.携帯電話向けブラウザの概要

1990年代末から、携帯電話キャリア各社は、コンテンツ・メールなどのインターネットアクセスサービスを開始しており、現在販売されているほとんどの携帯電話には組み込みブラウザ・メーラが搭載

*三菱電機 情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

されている。これらのサービスに対応するために、携帯電話向けに、我々は組み込みブラウザおよびメーラの開発を実施した(図1)。

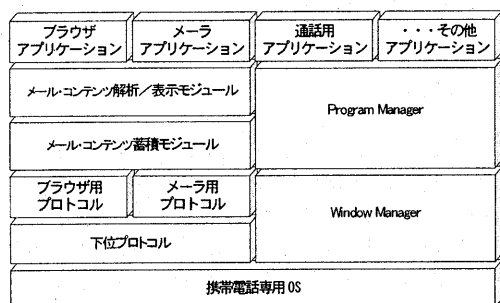


図1 携帯電話向けブラウザ周辺モジュール構成

一般PC(Personal Computer)向けブラウザと比較し、携帯電話向けブラウザは以下の課題を持つ。

(1)様々なコンテンツ記述形式・プロトコル

HTML(HyperText Markup Language)/HTTP(HyperText Transfer Protocol)などの標準規格が定められているPC向けブラウザに対し、携帯電話向けブラウザでは、キャリア各社毎に定められた独特のコンテンツ記述言語(基本はHTMLであるが、そのサブセットである場合が多い)やプロトコルに対応しなければならない。

(2)少ないハードウェアリソース

携帯電話向けソフトウェアは、ROM(Read Only Memory)・RAM(Random Access Memory)のサイズや、CPU(Central Processing Unit)の性能など厳しいハードウェアリソース制約を満たしつつ、要求機能の提供と十分なユーザレスポンスを実現しなければならない。

2.2.開発方針

開発規模が特に大きく、機能的に他の機種開発にも展開可能なコンテンツデータを解析・表示するモジュール:ブラウザエンジン(図1:メール・コンテンツ解析・表示モジュール)に着目し、流用による開発効率化を実現するために、下記の開発方針を設定した。

(1)コンテンツ記述言語に対する柔軟性

各キャリアのコンテンツ記述言語の基となっているHTMLを対象に、基本的なデータ構造・処理構造を構築し、タグやタグ属性の名前変更、機能の追加・削除に容易に対応可能とする。

(2)ハードウェアに対する柔軟性

当時の開発対象機種種のハードウェア制約(RAMサイズ・画面サイズ)にとらわれず、RAMが増減した場合、画面サイズが変更された場合にも容易に対処可能な設計を行う。サイズ変更による影響を考慮すべきデータタイプ・定数定義値を明確化し、これらの再定義を行えば基本データ構造・処理構造に影響を与えることなく変更が可能な実装を目指した。

2.3.開発の成果

携帯電話向けのブラウザエンジンとして、対象とするコンテンツの規模としては小さい(10K程度)が、使用するメモリ量の抑制が容易に行え、また拡張可能なブラウザエンジンモジュールを得た。

3.カーナビ向け組み込みブラウザ・メーラの開発

3.1.カーナビ向けブラウザの概要

携帯電話でのブラウザソフトウェアを流用し、カーナビ製品向けブラウザ・メーラとして、製品化開発を行った(図2)。一般的な市販カーナビにおいては差別化機能の一つとして通信機能が備えられ、また自動車メーカー各社においても地図と連動した位置情報コンテンツを整備し、インターネットを通してサービスを提供する動きが出てきており、カーナビにもインターネット接続機能が標準的に搭載されつつある。

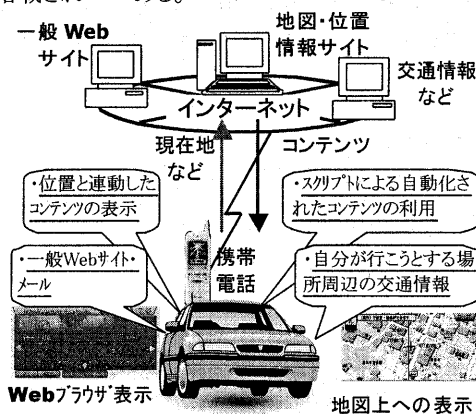


図2 カーナビ向けブラウザのシステム概略

カーナビ向けのブラウザは、インターネット標準のHTML/HTTPに準拠するだけでなく、車両や店舗の位置情報を記述する方式として標準化が進んでいるPOIX(Point Of Interest eXchange language)⁴⁾に対応した位置情報交換機能や、

JavaScript⁵⁾によるスクリプト処理機能を新規に追加した。

3.2. 携帯電話向けブラウザエンジンの流用

携帯電話向けブラウザエンジンをカーナビ向けブラウザの HTML 解析処理に流用することで、開発期間の短縮を図った。実際のソフトウェア開発では以下の項目を実施した。

(1) HTML3.2 相当のコンテンツ記述への対応

カーナビ用として流通しているコンテンツなどの対応 HTML バージョンを 3.2 相当であり、HTML3.2 要素セットに対する解析処理を実装した。これには、携帯電話向けブラウザエンジンが解析対象として持つコンテンツ各要素の情報として、HTML3.2 相当のものが最初から準備されていたことから、解析処理はほぼそのまま流用することができた。

(2) 処理可能コンテンツサイズの拡大

携帯電話向けブラウザは、使用可能 RAM リソースの制約から処理可能なコンテンツサイズにも制約をつけた実装が行われていた。カーナビでは、使用可能 RAM リソースが増えるため、コンテンツサイズ制約をなくす設計とした。開発ライン数 1 千行弱のメモリ管理処理の変更、データタイプのサイズ・定数定義値の変更を実施したが、基本的な内部データ構造は特に変更することなく流用可能であった。

(3) 描画処理のカーナビ向け開発

カーナビ実装環境は高機能な描画処理用コンポーネントモジュールが提供されている。カーナビ各アプリケーションのユーザインタフェースを統一するという前提のもと、ブラウザエンジンが解析して生成したコンテンツ要素データを描画コンポーネント向けのデータに変換する必要があった。携帯電話の描画処理構造は流用できず、新規開発となった。

(4) 位置情報記述解析への対応

新規開発であったが、位置情報記述言語 POIX が HTML に近い言語仕様を持つため、携帯電話向けブラウザエンジンの解析処理構造を流用した設計・実装を行うことができた。

3.3. 評価

開発規模実績で評価すると、携帯電話向けブラウザエンジンの 95% のソースコード流用が可能であった。これは、カーナビ向けブラウザエンジン全体の 55% にあたる。

また、開発工数実績で評価すると、カーナビ向けブラウザエンジンを新規に開発した場合の予測工数に対し、実際の開発では 63% の工数削減を

実現した見積もりとなる。ブラウザエンジン部では、各種タグ・タグ属性などの組合せ、不正記述コンテンツ、端末リソースを多く消費するコンテンツなど、多くのケースを考慮する必要があり、携帯電話向け開発では品質確保に大きな工数を要した。これに対しカーナビ向け開発では、次の2つの携帯電話向けブラウザ開発当初の設計目標達成により、品質の確保されたデータ構造・処理構造をほぼそのまま流用することができ、品質確保作業を大きく削減することができた。

- ・ 字句解析処理部分が HTML3.2 への対応だけでなく、位置情報記述に関する解析にも用いることができ、柔軟に要素セットの追加・交換が可能であったこと
- ・ メモリ制約・画面サイズが大きく変更されたカーナビ上でも、基本的なデータ構造・処理構造を変更する必要がなかったこと

4. 拡張容易性への取り組み

4.1. カーナビ開発における問題点

カーナビ向けブラウザ開発への流用を通し、以下の問題点が挙げられた。

(1) 実装環境の表示・操作処理部の相違

表示処理・操作処理部は、実装環境により大きく異なる。携帯電話開発当初は、実装環境それぞれの低レベル描画 API (Application Program Interface) 層で相違を吸収することを想定していたが、カーナビのように実装環境の描画コンポーネントを利用し、ユーザインタフェースを統一するという開発要件には対処できなかった。このためカーナビ開発では、携帯電話向けブラウザエンジン中の描画・操作処理依存部の調査・抽出作業を実施しなければならなかった。

(2) 処理可能なデータフォーマットの追加

コンテンツ中に存在する画像やサウンドなどの埋込みデータフォーマットは様々であり、要求仕様に応じ多様なデータフォーマットへの対応が求められる。実際カーナビ向けブラウザ開発では、携帯電話でサポートしていなかった画像データフォーマットや JavaScript など、コンテンツ埋込みデータの処理モジュールを追加した。携帯電話向けブラウザエンジンでは、埋込みデータ処理構造が独自のモジュールに依存した実装となっており、処理モジュールの追加において依存部の調査・抽出作業を実施しなければならなかった。

以上のカーナビ開発における問題点のフィードバ

ックを受け、更なる再利用性の向上を図るために、次節以降に述べるモジュール構造の再設計を試みた。

4.2.MVC(Model View Controller)設計手法の適用

MVC 設計手法⁶⁾とは、データ管理・表示・操作それぞれの機能を、独立したモジュール Model・View・Controller に分割、インタフェースを明確化し、それぞれの要求仕様変更に伴う作業を局所化する方法である。

組み込みブラウザへの適用において、各データフォーマット処理モジュールそれぞれを MVC 構造とした。これにより、実装環境の表示・操作系に依存する変更作業は、それぞれ View・Controller モジュールに集約することが可能である。また、機種毎の仕様相違が少ないデータ処理などの Model モジュールについては、手を加えることなく他の機種に横展開することができ、HTML系の View モジュールは、微妙に異なるコンテンツ記述の違いに対しても共通の処理として使用することができる。

4.3.組み込み向けプラグイン機構の設計

新しいデータフォーマット対応を容易化するため、個々のデータフォーマット処理部とモジュール統合部を分離し、相互間にいくつかのインタフェースを規定する。この規定インタフェースに則ったモジュールを作成することにより、容易に新データフォーマット処理を組み込んだり、不要なデータフォーマット処理を削除したりすることが可能

なプラグインモジュール機構を構築した(図3)。

データフォーマット処理部とモジュール統合部とのインタフェース規定にあたっては、将来的にインターネットからプラグインモジュールをダウンロードし、データ処理機能を自動拡張することのできる動的なプラグインを実現することを考慮に入れた。動的プラグインによる製品の機能向上を図ると共に、これまでの組み込みソフトウェア開発における大きな課題の一つであった、製品出荷後に発覚した障害対応への容易化を狙いとしている。

組み込み機器向けの動的プラグイン実現においては、ハードウェアリソース制約から以下の問題が存在する。

- ・ 組み込み機器では、プラグインモジュールの保存領域が小さく抑えられてしまう。このため、ダウンロードしたモジュールを単に追加保存していただくだけでは、簡単に領域不足を引き起こしてしまう。
- ・ 新しいプラグインを追加して実行するとき、CPU に与える負荷や必要なメモリサイズなどが分からない。このため、ブラウザが既存のプラグインと新規追加したプラグインを同時に実行できるかどうかの判定が難しい。

これらの問題に対し、次のインタフェース規定と機能を、各々のプラグイン処理部とプラグイン統合部に持たせることにより対処した^{2),3)}。

- ・ プラグインの実行に、呼び出し期限、呼び出し回数、処理データサイズなどの制限を設け、制限を超えたプラグインを自動的に削除す

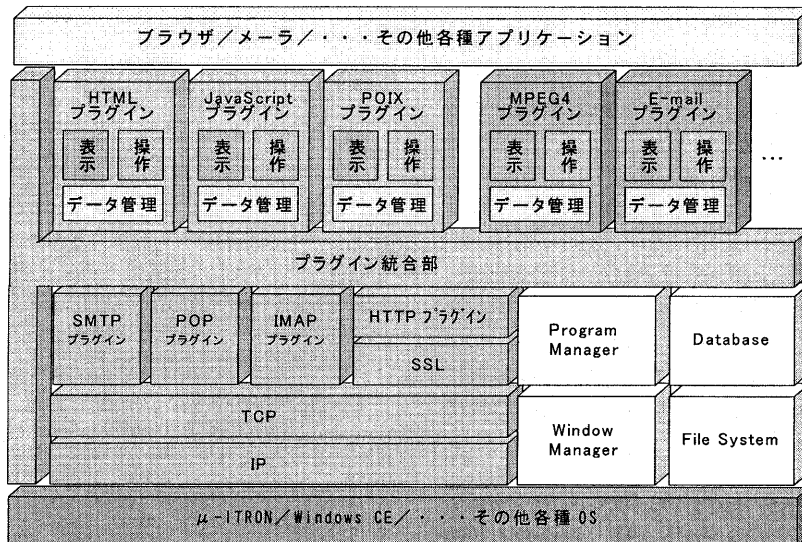


図3 再設計後の組み込み機器向けブラウザ構成

る。

- ・コンテンツ解析・表示時に必要・不要なプラグインを把握し、新しいプラグインのダウンロードが必要なときに保存領域が溢れてしまう場合には、不要プラグインを削除する(図4)。その際にプラグイン間の優先度、依存関係などを考慮する。
- ・各々のプラグインのある動作状態におけるメモリ・CPU 負荷状況通知手段、およびプラグインの動作状態変更手段についてのインタフェースを規定する。モジュール統合部側では、プラグイン側に設けたインタフェースを用いてメモリ・CPU 負荷を分散させることにより、プラグイン競合実行を実現する。

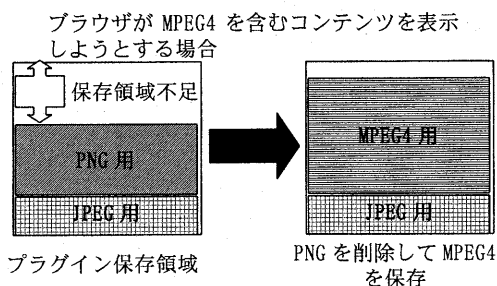


図4 プラグイン自動構成

また、2 次記憶が利用できる環境においては、プラグインモジュールを一旦 2 次記憶領域に置いておき、必要になった時点でローディングするなどの対応が行えるようになる。

これらの動的プラグイン機構はリソースの限定される組み込み機器向けソフトウェアにおいて有効であり、開発効率の向上とともに、機能的にスケラブルなブラウザを実現できると考える。

4.4.通信モジュールのプラグイン化

カーナビ上のブラウザ開発においては、携帯電話で用いられる通信方式が特有のものであり、TCP/IP などの一般的なプロトコルとは異なったものであったため、通信プロトコルモジュールの再利用はほとんど行えていなかった。

しかし、コンテンツ処理と同様に、今後は様々な通信プロトコルを利用したメディア処理が行われる事が予想され、組み込み機器においても必要とする通信プロトコルに対するプラグイン機構は有効であると考えられる。

プラグイン統合部には、コンテンツの場合と同様に、必要とするプロトコルの種別を判定する判

定処理部を設け、必要に応じてプロトコルモジュールを組み込み、通信処理を行う。また、各プロトコルモジュールは必要な通信を行うための共通の抽象化されたインタフェースを備え、上位のモジュールからは共通した呼び出しを行う事ができる。

5.まとめ

携帯電話向けに開発した組み込みブラウザエンジンをカーナビ向けに流用することにより、開発効率化を実現した。携帯電話向けブラウザエンジンの開発にあたっては、機種シリーズ化や他キャリア向けへの展開を目標に、ハードウェア性能やコンテンツ記述言語の相違に柔軟に対処可能な設計を目指したが、カーナビ向け開発で設計内容の有効性を実証できた。

また、カーナビ向けブラウザ開発で挙げられた問題点に対処し、MVC 設計手法導入によるデータ管理・表示・操作処理変更の容易化を行った。さらに、限定されたリソースにおいても、インターネット上の様々なデータフォーマットや通信プロトコル対応を可能にするプラグイン機構を持つブラウザ・メーラの設計を行った。

今後は、新設計による組み込みブラウザの実用化を進め、性能面での評価を行うと共に、プラグイン構造導入のもう一つの目標である、インターネットからプラグインモジュールをダウンロードし、データ処理機能を自動拡張する動的プラグインの実現を計画している。

- 1) 山中弘, 泊陽一郎, 吉田玲子: 携帯電話向け組み込みブラウザの開発, 第 60 回情報処理学会全国大会(Mar, 2000)
- 2) マット モラン: Embedded System Plugin Re-configuration Preference, 情報処理学会 FIT2002
- 3) 山中弘, マット モラン, 齋藤正史: 組込ブラウザにおけるプラグイン競合実行制御, 情報処理学会 FIT2002
- 4) モバイル標準化検討委員会(MOSTEC): POIX: Point of Interests eXchange language Version 2.0 (1999)
- 5) David Flanagan: JavaScript: The Definitive Guide, 4th Edition, O'Reilly (2001)
- 6) Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing(1994)