携帯電話向けのXML処理用ミドルウェア

益子由裕[†] 並木美太郎[†]

現在，iアプリ等の携帯電話上のJavaアプリケーションの開発において，Web上のコンテンツと連携した実用的なアプリケーションの実現については今後の課題となっている．一方で，Web上ではXMLが注目されており，今後さらにXMLによるコンテンツが増加すると推測される．そこで，本研究では携帯電話用Java上でのXMLの管理・解析を実現するためのミドルウェアを開発し，これらWeb上のコンテンツを活用したアプリケーションの作成を支援する．しかし，携帯電話はCPU・メモリともに低性能であり，フルセットのXML文書を扱うソフトウェアの開発が困難である．そこで，XMLをサブセット化したcompactXML，SAX1.0をサブセット化したcompactSAXを策定した．本仕様は，フルセットのXMLの仕様からDTDおよび実体参照を削除したXML宣言，要素，CDATAセクション，処理命令等の仕様によりデータを記述する枠組みを提供する．本仕様に基づいたXMLパーサとしてcSAXとファイル管理のミドルウェアを携帯電話上に実現した結果，cSAXを約6KB，ファイル管理を約8BK程度の大きさで実現でき，省メモリのミドルウェアを開発することができた．

XML Middleware for Cellular Phones

Yoshihiro Mashiko[†] and Mitaro Namiki[†]

Now, in development of the Java application on cellularphones like i-appli, it has been a future subject about realization of the practical application which cooperated with the contents on Web. On the other hand, on Web, XML attracts attention and it is surmised that the contents by XML will increase from now on further. Then, in this research, the middleware for realizing management and analysis of XML on Java for cellularphones is developed. And the creation of application which utilized the contents on these Web is supported. But, both a CPU and a memory are low performance, and the development of the software which handles the XML document of the full set is difficult with cellularphones. Then, it decided upon compactXML which subset-ized XML, and compactSAX which subset-ized SAX1.0. Small middleware was able to be developed as a result of using such specifications.

1. はじめに

現在，携帯電話はiモード等のWeb接続サービスによって，最も手軽な情報端末としての地位を確立している．そして，新しい携帯電話にはiアプリ，ezplus，Javaアプリといった Java実行環境（以下，携帯電話用Java実行環境）が搭載されている．携帯電話用Java実行環境上で動作するアプリケーションプログラム（以下AP）ではHTTP（S）通信がサポートされており，Web上のコンテンツと連携したアプリケーションを作成することができる．また，現在の携帯電話用Java実行環境では，発売当初に比べアプリケーションサイズやデータストレージの許容値が拡張され，当初に比べ大きいアプリケーションの実現が可能となっている．さらに，インターネットへの接続やネイティブアプリケーションへのアクセス等に関する制限が緩和され，より実用的なAPの実現も可能となっている．

しかし，現在の携帯電話はカメラの搭載といったAV分野の強化からも分かるように，エンタテイメント方向へ視野が向いている．携帯電話用Java実行環境上で利用されているAPの多くもスタンドアロン型のゲーム等が中心で，日常的に利用されるような実用的なAPは数少ない．特に，他のPDAといった情報端末のような高機能な処理や，Web上のコンテンツと連携した実用的なAPの実現については今後の課題となっている．これは，携帯電話用Java実行環境上のAPとWeb上のコンテンツとを効果的に連携するフレームワークが確立されていないことが原因として考えられる．

一方，Web上の多くのコンテンツはXMLベース

† 東京農工大学工学研究科
Graduate School of Technology, Tokyo University of Agriculture and Technology

のシステムへと移行が進められている．汎用的なデータ形式として，XML が Web 上で広く利用され始めているのは周知の事実である．なかでも特に，データとデザインの分離という XML の特性を活かし，クライアントやサーバで XML プロセッサを用いたシステムが利用されている．

つまり，携帯電話用 Java 実行環境上で XML を利用するフレームワークを提供することで，Web 上の XML により記述されたコンテンツを活用する AP の作成が可能になると考えられる．

2. クライアント指向型のフレームワーク

2.1 概　　要

実用的なモバイルアプリケーションの実現を考えると，ネットワークを介して企業のデータベースやバックエンドのビジネス AP 等の Web 上の別の AP とデータの交換や共有を行うことになることが予想できる．独自のデータフォーマットを用いた場合は，データの汎用性が低下し，データが携帯電話内部だけで利用されるものとなる可能性がある．また，モバイルコンピューティング分野の企業によって，住所録や予定表といったユーザの個人情報管理のデータベースを特定の仕様に依存しない形式でデータの同期ができるように SyncML という XML ベースの規格が近年定められた[5]．このように，モバイルアプリケーション上で XML を活用する動きが見られ，今後はこの動きがさらに加速すると考えられる．したがって，モバイルアプリケーションで XML を活用することで，AP の柔軟性の向上や開発コストの軽減だけではなく，データの汎用性や再利用性を大幅に向上させることが可能である．特に，データの相互運用性を確保するために，Web 上の標準的なデータ表現形式である XML を用いる利点は大きい．

Web 上で XML を利用する場合，サーバ側かクライアント側のどちらかで XML プロセッサによる XML 文書の解析が必要となる．現状において，XML を用いた一般的なシステムでは，サーバ側で XML 文書の解析を行うことが多い．特に，携帯電話等の低性能な組み込み機器をクライアントとしたシステムで XML を利用する場合には，XML プロセッサ自体の容量や XML 文書の保持や解析における負荷等を考慮し，サーバ側で XML 文書の解析を行うことが一般的である．

しかし，現在の携帯電話では赤外線通信等によるデータ交換[1] が可能になり，携帯電話内のデータを活用する場面も増えてきている．特に，携帯電話は常時持ち歩き，個人情報等のパーソナルなデータが管理されるという特性がある．この特性から今後，携帯電話はポータブルかつパーソナルなデータベースとして活用されると考えられる．たとえば，モバイルアプリケーション側に保存されている個人情報を使って，オンラインストアから旅行の予約や本の購入といったモバイル電子商取引等の局面での利用が可能となる．また，モバイルアプリケーションどうしで赤外線を通じて，名刺等の情報交換を行うことや，モバイルアプリケーション対 PC で赤外線通信によりデータ交換することも考えられる．こういった場合，汎用的なデータフォーマットである XML を利用することは効果的であり，モバイルアプリケーション側でその XML の管理・処理を行うことが望ましい．

たとえば，サーバ側やプロキシを介して XML の解析を行い，その解析結果を HTML 形式によってモバイルアプリケーションに渡すフレームワークを利用する方法も考えられる．しかし，このフレームワークだとモバイルアプリケーション利用時につねにインターネットへ接続する必要があり，携帯電話が圏外である場所では，モバイルアプリケーション自体が利用できなくなるという致命的な欠点がある．携帯電話側で XML 以外のデータ形式で管理しておく方法もありうるが，この方式ではサーバ側に AP ごとのデータ形式の変換を用意する必要があり，サーバプログラムの開発の手間が多いこと，データの相互運用性が乏しくなる欠点がある．

クライアントである携帯電話のモバイルアプリケーション側で XML 文書の管理・解析を行うことで，管理している XML 文書に対し，データの形式が定義されたタグを利用した多様な処理を行い，携帯電話をパーソナルでポータブルな XML データベースとして有効活用することが可能になる．また，保持している XML 文書を解析することで，携帯電話が圏外である場合も含め AP が常時利用可能になるという利点を享受できる．さらに，インターネットへのアクセス回数を削減できるので，パケット料を抑えられるとともに，XML データベースを利用したパーソナルなデータベース等を，スタンドアロン型の AP として開発できる．データ形式として XML ないしはそのサブセットとすることで，XML データを多くの Web 上の AP でそのまま，ないしは簡単な変換で利用することができるようになる．

そこで，本研究では，クライアントであるモバイルアプリケーション側で XML の管理と解析を行う，クライアント指向型のフレームワークに焦点を当てる．これによって，Web 上の XML により記述されたコ

ンテンツを活用する AP の作成が可能になると考えられる．

2.2 問 題 点

クライアント指向型のシステムを実現するうえで問題となるのは携帯電話用 Java 実行環境上で動作する XML 処理ソフトウェア，特に基本となる XML パーサの有無である．XML パーサは，XML 文書の解析を携帯電話用 Java 実行環境上で実現するためには必要不可欠なものである．本研究では，このクライアント指向型のフレームワークを提供するうえで，i アプリ（504i 以降のシリーズ）をターゲットデバイスとした．これはデータストレージである Scratchpad の構造やその容量，そして，HTTP 通信の利用制限等の問題からである．504i シリーズの AP のサイズの許容値は JAR ファイルで 30 KB，Scratchpad の許容値は 100 KB であり，データストレージについては他の仕様に比べ余裕があるが，アプリケーションのサイズは小さい．他の JavaVM を搭載している携帯電話も同様のハードウェア仕様を有しているが，いずれにせよ，メモリサイズは少なく，CPU 能力は低い．少なくとも通常の Java 2 Standard Edition 上の Java アプリケーション上で利用されている Crimson（Java 2 SDK 1.4 系にバンドルされている XML パーサで容量は約 220 KB）[6] や Xerces（Apache XML プロジェクトの XML パーサで容量は約 1.8 MB）[7] といった一般的な XML パーサはプログラムサイズがきわめて大きく，携帯電話用 Java 実行環境上で実行することはできない．

プログラムサイズが小さい小型の XML パーサとして，Min（パーサ容量は 28 KB）[8]，TinyXML（パーサ容量は 23 KB）[9] 等が存在しているが，これらの XML パーサも Java 2 Micro Edition をターゲットとしたものではなく，携帯電話用 Java 実行環境上では利用できない．また，i アプリの 30 KB という AP のサイズの許容値を考えると，十分縮小化されているとはいえない．TinyXML は DTD 等の機能もサポートしており，フルセットの XML 文書を扱えるパーサであり，これ以上の縮小化は望めない．Min は，XML の機能が非常に限定され，要素のみしかサポートされておらず，属性や混在内容といった通常の XML 文書内でよく利用される機能もサポートしていない Minimal XML を扱うパーサである．しかし，通常の XML 文書との互換性が大きく失われているにもかかわらずプログラムサイズは大きい．

現在の i アプリの仕様では，複数の i アプリ間におけるデータやプログラムの共有はできない．したがって，XML パーサを多数の i アプリから利用される共有のライブラリという形で端末に導入することはできず，つねにその XML パーサを利用する i アプリ内に同梱する必要がある．さらに，低処理能力である i アプリ上での XML 文書の解析処理については，パフォーマンス面で大きな負荷となり，AP の実行速度に影響を与えることが予想できる．

つまり，クライアント指向型のフレームワークには，携帯電話用 Java 実行環境上で動作し，十分にプログラムサイズが小さく，オーバヘッドの少ない XML パーサが必要となる．そして，その XML パーサの実現にあたっては，フルセットの XML では仕様が大きすぎるという問題がある．今後，携帯電話のハードウェアはより高性能になると考えられるが，格納されるデータの領域は今後もさらに増加傾向にあり，不要な機能を削除したサブセットの意義は現在も将来も十分意義があると考える．

また，クライアント指向型のフレームワークでは，クライアントである i アプリ上に XML 文書を保持する必要があり，データストレージである Scratchpad の重要度は高い．しかし，Scratchpad にはファイルシステムが存在しないため，プログラマ自身がファイル情報を詳細まで管理する必要があり，XML 文書等のデータ管理が行いにくい．そこで，本研究では，XML 処理系とともにファイル管理を行うミドルウェアも必要であると考えた．

3. 目 標

本研究では，クライアント指向型のフレームワークを提供するために，携帯電話用 Java 実行環境（Java 2 Micro Edition）上での XML 文書の管理・解析を実現するミドルウェアを開発することを目標とし，具体的に以下の 3 点を実現する．

(1) compactXML の策定
(2) 携帯電話用 Java 実行環境用の XML パーサライブラリの開発
(3) Scratchpad 用のファイル管理ライブラリの開発

前述したように，携帯電話用 Java 実行環境上ではフルセットの XML の仕様は大きい．そこで，本研究ではまず XML を携帯電話用 Java 実行環境に適合させるために，(1) において XML をサブセット化する．本研究では，この XML のサブセットを compactXML としている．次に (2) として，この compactXML に対応した携帯電話用 Java 実行環境用の XML パーサライブラリを開発する．compactXML は携帯電話用 Java 実行環境に対して必要最低限と考えられる基礎

的な機能のみで構成されており，これによりメモリやAPの実行速度への負荷を抑える．そして(3)として，Scratchpad上でのXML文書の管理を簡単にするためのファイル管理ライブラリを開発する．XMLパーサとファイル管理ライブラリにより，クライアント指向型のフレームワークを提供する．

また，携帯電話用Java実行環境は，より多機能・高性能化されてきているが，それでもPC等の実行環境に比べればメモリが少なく処理速度が遅い．このため，ミドルウェアの開発にあたっては，より省メモリでオーバヘッドが少ない必要がある．したがって，本研究ではこれを設計方針とし，compactXML同様に開発するミドルウェアは必要最低限の機能で構成し，無駄な機能はいっさい省くことで省メモリやオーバヘッドの減少を図っている．

省メモリ化に対する具体的な目標値として，各ライブラリの容量はそれぞれJARファイルで10 KB以下を想定している．10 KBというのは，504i以降のシリーズにおけるAPのサイズの許容値の3分の1である．したがって，ミドルウェア全体の目標値としては，20 KB以下を想定している．処理速度に関しては，解析するXML文書の大きさ等も影響してくるので一概にはいえないが，ユーザが不快感を感じない程度の実用に耐えうる処理速度であることを想定している．具体的な目標値としては，1 KB以下のXML文書ならば1秒以下で解析できることを想定している．

4. 構　　成

本ミドルウェアは，**図1**に示すように，XMLパーサライブラリである“cSAX”とファイル管理ライブラリである“FML”の2つのライブラリから構成される．cSAXはiアプリ上でのXMLの解析を実現するライブラリであり，実際にcompactXMLの解析を行う．そして，FMLはScratchpad上でのXMLファイルの管理を実現するライブラリであり，XMLファイル以外のファイルも扱えるファイルシステムをScratchpad上に構築する．

また，APによってはプログラムサイズを抑えること等を目的として，どちらか片方のライブラリのみを利用したい場合もある．これに対処するために本ミドルウェアは各ライブラリがそれぞれ単体で利用できるようにする．つまり，1つのXMLファイルしか利用しないため，ファイル管理ライブラリは必要ないという場合は，XMLパーサライブラリだけを利用するといったことも可能である．

cSAXはCLDC[9]上に構築されており，CLDCをサポートするデバイス上ならiアプリに限らず動作するが，FMLはScratchpadというiアプリ固有のデータストレージをターゲットとしているため，他の実行環境上では動作しない．

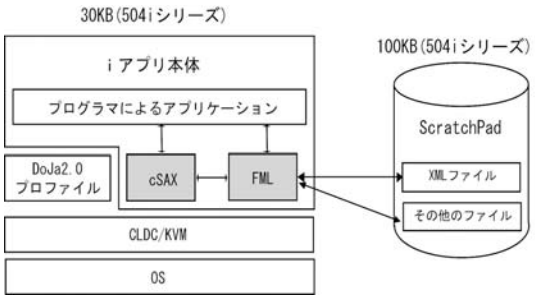

図1 本ミドルウェアの構成
Fig. 1 The composition of middleware.

5. compactXML

5.1 概　　要

プラットフォーム独立性や相互運用性を考慮すると，フルセットのXML文書を扱えることが理想的である．しかし，携帯電話等の組み込みデバイス上でXML文書を扱う際には，リソースやネットワークの利用に制限等があるため，フルセットのXML文書において定義されている機能のすべてが十分に使えない場合もある．また，フルセットのXML文書には必要不可欠とはいえない機能も定義されている．これらの不要と思われる機能を削減し，XMLをサブセット化することで，実装を単純化することが可能である．モバイルAP上でXMLを利用する場合には，メモリに対する負荷を軽減し，解析処理におけるパフォーマンスを向上させる必要があるため，XMLをサブセット化する意義があると考えられる．

本フレームワークではWeb上のシステムとXMLによるデータの交換・共有を行う．そこで，サブセット化するとはいえフルセットのXML文書との互換性については考慮する必要がある．つまり，互換性を損なわないようにXMLの構造を構成する基礎的な機能についてはサポートし，削減しても互換性が損なわれない機能については削減するという方針をとった．本フレームワークではこのXMLのサブセットをcompactXMLとして策定した．

5.2 仕 様 策 定

XMLのサブセットである以上，基礎的な構造はXMLと同じである必要があるし，Web上のコンテンツ等との相互運用性を考えても，XMLの基礎的な構造についてはサポートする必要がある．XMLで定

義されている最も基礎的な構造は要素（element）である．この要素は開始・終了タグと文字データによって構成され，開始タグには属性を含む場合もある．したがって，compactXML では基礎的な構造であるこれらの機能をサポートする．また，XML 文書であることの宣言文である XML 宣言についても必須の機能だと考えられる．さらに，HTML 等の文書指向型の XML を処理するうえでは，混在内容も必須の機能といえる．同様に，空要素も HTML 等の XML 文書を処理するうえで必要となる機能である．「<」や「&」といった XML において特別な意味を持つ文字を扱ううえで必要とされる定義済み実体や CDATA セクションといった機能も必要になると考え，これらの機能についてもサポートする．処理命令やコメントといった機能は必須とは考えられないが，これらの機能を削減したところで，XML パーサの省メモリ化には貢献できない．逆に，削減することは互換性を損ねるだけと考え，compactXML ではこれらの機能を提供している．

フルセットの XML 処理系と互換性を保つうえで必要最低限とされる機能とは反対に，XML の構造において必要不可欠とはいえず，サポートしなくても XML の本質に問題がないと考えられる付加価値的な機能がある．DTD はデータの文書構造や整形式チェックに有用であるが，構造そのものはデータにタグとして記述されており，このタグや属性の構造を AP から利用することが最低限の機能であると判断した．AP は，DTD がなくても XML の処理を行えるからである．フルセットの XML からは，単に DTD の削除ないしは簡単なプログラムで DTD のない形式に変換できる．また，DTD については宣言方法が適切ではないうえに，設定項目等が少なくスキーマ言語としての機能を果たしきれていないのが現状である．その結果，DTD は今後 XMLschema [12] や Relax [13] といった新しいスキーマ言語に置き換えられようとしている．つまり，現状で DTD をサポートしても利点はなく，DTD をサポートしないことによる XML プロセッサの省メモリ，高性能という利点の方が大きい．実体参照は DTD の中で参照すべき実体を定義するため，DTD をサポートしないと実体参照も利用できない．しかし，実体参照は HTML の画像を扱うように属性に参照先を指定し，AP 側で処理することで代用が可能であるし，むしろこの方法で利用されることの方が多い．したがって，DTD や実体参照といった付加価値的な機能については compactXML ではサポートしていない．

表 1 に compactXML で提供する機能と提供しない機能をまとめた．また，compactXML 文書に対応した例を**図 2** に示す．この仕様により，住所録をはじめとする個人情報の管理を行う XML データベースのエンジンや各種ユーティリティを作成できる．

表 1 compactXML の機能
Table 1 The functions of compactXML.

提供する機能	提供しない機能
XML 宣言	DTD
要素	実体参照
（開始・終了タグ/文字データ）	（外部内部実体）
空要素	
属性	
混在内容	
定義済み実体	
CDATA セクション	
処理命令	
コメント	

```
<?xml version="1.0" “encoding="Shift-JIS”?>
<root>これは混在内容
    <elem att=“1”>これは要素と属性</elem>
    <elem> &amp これは定義済み実体</elem>
    <elem><![CDATA[これは CDATA セクション]]></elem>
    <empty_elem/>これは空要素—
</root>
```

図 2 compactXML の一例
Fig. 2 A sample of compactXML.

フルセットの XML の定義においては，整形式・妥当性検証として，XML を解析する XML プロセッサの動作について言及している．妥当性検証は DTD を用いて行われるものであるが，フルセットの XML 用プロセッサでも必須の機能とはされていない．compactXML では DTD 自体を提供しないため，compactXML 用プロセッサでは妥当性制約については考慮しない．また，整形式制約の中でも DTD や実体参照に関連する制約があるが，これも compactXML では考慮しない．しかし，文字の範囲（Char），名前文字（NameChar），名前（Name）の定義といった基本的な整形式制約については，compactXML でも同様の制約を設ける．したがって，compactXML では DTD や実体参照等の削減された機能以外に関連する整形式制約については，守らなければならない．しかし，compactXML 用プロセッサでは，整形式検証は行うことが望ましいが必須ではないとする．

以上の理由から，compactXML 用の XML プロセッサは妥当性・整形式の両方の検証についてサポートしなくてもよいと考えた．前述した他の小型 XML パーサも妥当性・整形式検証について完全にサポートしているものはない．これは，パーサ容量の縮小化や解析処理におけるパフォーマンスの向上を図るうえで効果的であるからである．compactXML もこの見地から

策定しているため，妥当性・整形式検証についてサポートしない．

以上の仕様で，小型軽量な XML のデータベースエンジン等を記述するのに十分な仕様であると判断した．ただし，フルセットの XML 仕様のデータとの互換性において，次の 2 つの問題が存在しており，それぞれ次のように対処する．

第 1 の問題として，compactXML では DTD についてサポートしていないため，文書型定義の宣言である"<!DOCTYPE"で始まるタグを含んだ XML 文書については解析できない．そこで，compactXML 用プロセッサでは，文書型定義の宣言タグを読み飛ばして対処することとした．compactXML 用プロセッサでは，DTD における妥当性検証や実体の解決といった処理は行わないが，DTD を含んだ XML 文書についての解析については行う．これにより，compactXML とフルセットの XML の互換性が損なわれずに済む．

第 2 の問題は，実体参照を用いた XML 文書を扱う場合である．compactXML 用パーサでは解析時に実体を参照するマーク付けがあったとしても，DTD を読み飛ばしているため，その実体を解決することができない．先にも述べたが，実体参照については，リンク先のデータとなることが多く，ネットワークが接続できない状況下においてはは他マシンにあるデータを参照できないことになる．したがって，仮に本機能を導入したとしても，その機能を十分生かせないことがあるため仕様から削除した．ただし，実体参照については，参照名を利用できるようにした．必要ならば，リンク名を通常のデータと同様に AP で管理できる設計となっている．

6. cSAX

cSAX は i アプリ上での XML の解析を実現するライブラリである．cSAX における解析時のインタフェースには，イベント駆動型である SAX を compactXML に対応させてサブセット化した compactSAX を策定し，採用している．

6.1 API の選択

XML 文書の解析にあたっては，標準とされる 2 つの代表的な API である DOM と SAX が存在する．それぞれに特徴があり，一般に用途によってプログラマは適切に使い分けている．DOM の場合，XML 文書に対しランダムアクセスが可能で，XML 文書の構造を変更するといった場合に有効である．しかし，XML 文書を一時にすべて読み込んで構文解析し，主記憶上に木を展開するため，実行時のメモリ使用量が多く，解析時間がかかることが一般的に知られている．また，複雑な処理が可能な反面，パーサ自体のプログラムサイズが大きいといった問題がある．

一方，SAX はイベント駆動型の API であり，DOM のように処理に先立って XML 文書を全部読み込み，メモリ上に木を作成しない．先頭から順に XML 文書を読み込みながら解析を行い，“要素の開始” や “要素の終わり” といったタグを読むとイベントを生成し，AP にイベントを通知する．したがって，SAX は DOM に比べ実行時のメモリ要求も少なく，高速に動作する．また，XML 文書の更新や構造の変更といった複雑な処理はできないが，パーサ自体のプログラムサイズが DOM に比べ小さいという利点がある．

i アプリのメモリや実行速度といった実行環境を考慮すると，採用する API としては DOM ではなく SAX が適切であるといえる．また，SAX のバージョンとしては SAX1.0 を採用した．

表 2 正規の SAX1.0
Table 2 The regular SAX1.0.

org.xml.sax		
	Interface	
		AttributeList
		DTDHandler
		DocumentHandler
		EntityResolver
		ErrorHandler
		Locator
		Parser
	Class	
		HandlerBase
		InputSource
	Exception	
		SAXException
		SAXParseException
org.xml.helpers		
	Class	
		AttributeListImpl
		LocatorImpl
		ParserFactory

6.2 compactSAX

前節の理由から，本ミドルウェアにおける XML 文書の解析時の API として SAX を採用したが，本ミドルウェアの解析対象はフルセットの XML 文書ではなく，XML のサブセットである compactXML となる．前述したように，compactXML はサブセット化により機能が限定されているため，SAX も compactXML に対応させる形でサブセット化する必要がある．本研究では，この SAX1.0 をサブセット化させた API を compactSAX とした．

表 2 に示すように，正規の SAX1.0 は Java で 7 つ

表 3 compactSAX
Table 3 compactSAX.

compactSAX		
	Interface	
		AttributeList
		DocumentHandler
		ErrorHandler
		Parser
	Class	
		HandlerBase
		InputSource
	Exception	
		SAXException

のインタフェース，2 つのクラス，2 つの例外によって定義されている．また，ヘルプパッケージに 3 つのクラスが用意されている．compactSAX では，これらの中から compactXML でサポートしていない機能を提供するインタフェース，クラス，例外を削減している．

インタフェースについては，DTDHandler，EntityResolver，Locator インタフェースを削減している．DTDHandler，EntityResolver インタフェースは DTD や実体参照関連の機能を定義してあるが，これらの機能は compactXML でサポートしていないので必要ない．つまり，DTD や実体参照関連の機能についてはイベント化しない．Locator インタフェースは，解析時の行番号やカラム数といった情報を提供する．これは，エラー発生時にそのエラーがどこで発生したかを通知する場合等に利用されるが，たとえこれらの情報を通知したとしても，携帯電話上において XML 文書を閲覧するエディタがなく，そのエラー個所を直す作業を行うことはない．したがって，compactSAX には Locator インタフェースは必要ない．これにともない，Locator インタフェースに定義されている情報を利用して例外を発生させる SAXParseException 例外についても削減している．さらに，ヘルプパッケージについてはそれらのクラスはなくても解析機能には何の問題もないため，ヘルプパッケージ内のクラスはすべて削減している．

その結果，compactSAX は**表 3** に示す 4 つのインタフェース，2 つのクラス，1 つの例外によって定義されている．また，DTDHandler，EntityResolver インタフェースを削減したことで，Parsr インタフェース内に定義されている setEntityResolver，setDTDHandler メソッドも不要となるため，これらのメソッドについても削除してある．したがって，compactSAX における Parser インタフェースは，SAX1.0 における Parser インタフェースのサブセットになっている．表 3 に示した各インタフェース，クラス，例外の概要は次節で示すが，これらは compactXML を解析するうえで必要となるものであり，compactXML で定義した機能のすべてに対応している．compactSAX では compactXML でサポートしている要素の開始・終了タグや文字データ等の基礎的なイベントやエラー関連のイベントのみを通知する．**表 4** に compactSAX における通知・非通知のイベントをまとめた．このように，compactSAX では SAX1.0 の機能をよりコンパクトにすることで，パーサ自体の容量の縮小化を図っている．同時に，これによって解析処理が単純化されてオーバヘッドの削減もできる．

表 4 compactSAX におけるイベント
Table 4 The events about compactSAX.

通知	非通知
startDocument	resolveEntity
endDocument	notationDecl
startElement	unparsedEntityDecl
endElement	
character	
error	
fatalError	
warning	
processingInstruction	

6.3 クラス構成

cSAX のクラス構成は表 3 に示した compactSAX のインタフェース，クラス，例外に実際に XML 文書の解析を行うクラスである cSAXParser クラスを加えた，以下の 4 つのインタフェース，3 つのクラス，1 つの例外によって構成されている．

- AttributeList インタフェース
  属性情報に関する扱いを定義したインタフェースで，実際のメソッドの内容は cSAXParser クラスに実装している．
- DocumentHandler インタフェース
  startDocument や character といった要素の開始・終了タグや文字データ等の基礎的なイベントのコールバック用メソッドを定義しているインタフェース．
- ErrorHandler インタフェース
  error や warning といったエラー関連のイベントのコールバック用メソッドを定義しているインタフェース．
- Parser インタフェース
  イベントハンドラの登録や実際に解析を開始する際に発行する Parse メソッド等を定義しているインタフェースで，実際のメソッドの内容は cSAXParser クラスに実装している．

- HandlerBase クラス
  DocumentHandler，ErrorHandler インタフェースを実装し，解析時において実際にイベントの通知を受け取るクラス．プログラム作成時にベースとなる．
- InputSource クラス
  解析対象となる XML 文書からの入力を管理するクラス．
- cSAXParser クラス
  AttributeList，Parser インタフェースを実装し，実際に compactXML 文書の解析を行うクラス．このクラスが cSAX の中心となる．
- SAXException 例外
  compactSAX におけるエラー全般をカプセル化するクラス．

実際に cSAX ライブラリを利用する場合には，HandlerBase クラスを継承するか，直接 DocumentHandler，ErrorHandler インタフェースを実装したクラスを作成する．そして，そのクラスで cSAX イベントを受け取り，その中のコールバック用メソッドを適当な処理でオーバライドしていくことになる．

7. FML（File Management Library）

前述したように Scratchpad にはファイルシステムが存在しない．そのため，プログラマ自身がバイト単位でファイルやデータの格納領域を指定する必要があるとともに，そのアドレス情報やファイルの属性情報等についても管理する必要があり，快適に利用できる環境ではない．FML はこれを解決するライブラリであり，Scratchpad での XML 文書の管理を簡単に実現する．また，FML は XML 専用のデータベースという形ではないので，XML ファイルだけではなく，その他のファイル全般の管理が可能であり，ファイル管理ライブラリとして FML 単体でも利用することができる．**図 3** に示すように，FML では Scratchpad 上に FileManager というファイル管理システムを構築し，ファイルを管理する．

また，FileManager は「System Property」,「File Entry Pool」,「Data Area」の 3 つのコンポーネントから構成される．各コンポーネントは以下に示す役割を持つ．

- System Property
  FileManager のサイズ等，システムのメタデータを格納する領域．
- File Entry Pool
  FileManager 内に管理されているファイルの属性情報等のメタデータを格納する領域．
- Data Area
  実際にファイルの内容をブロック化して格納する領域．

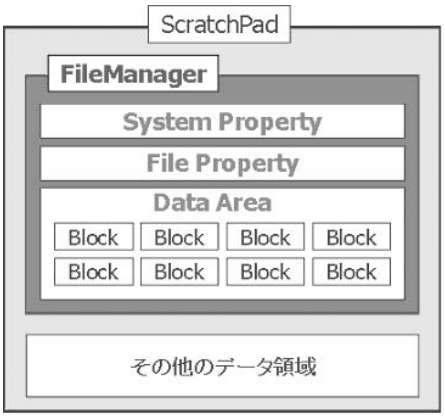

図 3 FileManager の構成
Fig. 3 The composition of FileManager.

7.1 機能概要

FML はライブラリ容量を抑えるために，必要最低限の機能で構成されている．このため，提供する機能はアドレス情報の管理，属性情報の管理，入出力の管理の 3 つである．各機能の概要を以下に示す．

- アドレス情報の管理
  ファイルを格納する際にデータを各ブロックに割り当てる．また，割り当てたブロック情報を管理する．
- 属性情報の管理
  ファイル名やファイルサイズといったファイルの属性情報を管理する．
- 入出力の管理
  Scratchpad に対しネイティブなメソッドを使ってアクセスするのではなく，FML で用意した入出力用のクラスを用いてファイルの入出力を行う．

これらの機能を提供することにより，プログラマはファイル情報やファイルを格納するアドレス情報の管理といった煩わしさから解放される．特に，複数のファイルを扱う場合や cSAX と併用する場合に効果的である．FML を用いると，compactXML で画像ファイル等を属性として利用する場合に，Scratchpad 上のアドレスではなく，ファイル名を記述することで，ファイルを指定することが可能となる．

また，上記 3 つの機能以外の機能については FML では提供しない．しかし，Scratchpad と i アプリには 1 対 1 の対応関係があり，Scratchpad 内のデータへは対応する i アプリからしかアクセスできないとい

う仕様上，セキュリティについては必要ない．そして，トランザクションについても Scratchpad のネイティブなトランザクション機能を利用するため必要ない．また，たかだか 100 KB 程度のデータストレージ上のファイルシステムなので，ディレクトリ構造を持たないフラットなファイルシステムで十分であると考えられる．

7.2 クラス構成

FML は前節であげた 3 つの機能を実現するために，以下に示す 6 つのクラスを用意した．

- FileManager クラス
  FileManager の作成，開閉等の FileManager に関する基本的な操作全般を司るとともに，System Property を管理する．
- File クラス
  ファイルの属性情報やアドレス情報をカプセル化するとともにそれらの情報を管理する．
- FileInputStream クラス
  InputStream クラスをベースとし，バイナリファイルからの入力を管理する．実際には，read() メソッドを用いてファイルからデータを読み込む．
- FileOutputStream クラス
  OutputStream クラスをベースとし，バイナリファイルへの出力を管理する．実際には，write() メソッドを用いてファイルへデータを書き出す．
- FileReader クラス
  InputStreamReader クラスをベースとし，テキストファイルからの入力を管理する．実際には，read() メソッドを用いてファイルからテキストデータを読み込む．
- FileWriter クラス
  OutputStreamWriter クラスをベースとし，テキストファイルへの出力を管理する．実際には，write() メソッドを用いてテキストファイルへデータを書き出す．

FML は上記 6 つのクラスから構成されており，ライブラリ容量を抑えるため，FML 独自の例外クラス等は用意していない．FML 使用時に発生した例外は，すべて CLDC における IOException を例外処理として throw するようになっている．

実際に，FML を利用する場合は，まず FileManager クラスの create() メソッドを用いて，Scratchpad 上に FileManager を作成する必要がある．その後ファイルを作成し，各クラスを用いてファイルの属性情報を得て，ファイルに対する入出力処理を行うことになる．

表 5 cSAX 内の各クラス容量
Table 5 Class capacity about cSAX.

クラス名	容量（.class）	容量（.jar）
AttributeList	約 0.3 KB	約 0.3 KB
DocumentHandler	約 0.5 KB	約 0.5 KB
ErrorHandler	約 0.3 KB	約 0.3 KB
Parser	約 0.3 KB	約 0.3 KB
HandlerBase	約 0.9 KB	約 0.6 KB
InputSource	約 0.9 KB	約 0.6 KB
cSAXParser	約 5.4 KB	約 2.5 KB
SAXException	約 0.7 KB	約 0.5 KB

```
<?xml version="1.0" encoding="Shift-JIS"?>
<connection id="識別子">
 <url>接続先 DB の URL</url>
 <driver>ドライバのクラス名</driver>
 <user>ユーザー名</user>
 <password>パスワード</password>
 <property name="プロパティの値"value="プロパティの値"/>
</connection>
```

図 4 JDBC の接続設定を XML で記述するサンプル
Fig. 4 A XML sample about JDBC.

8. 実装と評価

8.1 cSAX

現版の cSAX については，表 1 で示した compactXML がサポートする機能の中の CDATA セクションや定義済み実体については未対応であるが，それ以外の機能についてはすでに実装済みであり，基本的な解析処理は行うことが可能である．**表 5** では，この版の cSAX についての詳細を示してある．この表からも分かるように，cSAX 全体の JAR ファイルのサイズは約 6.0 KB 程度となる．今後，CDATA セクションや定義済み実体に対応させることで，多少はライブラリ容量が増加するが，目標値である 10 KB 以下には抑えられている．この結果から，cSAX のライブラリ容量については，十分に省メモリ化していると考えられる．

また，cSAX の動作確認も含め，XML 文書を cSAX を用いて解析する簡単なサンプル AP を作成した．このサンプル AP は，XML 文書を読み込んでその要素の内容や属性の内容を表示するもので，**図 4** に示す XML 文書を cSAX に読み込ませた．**図 5** に NTT ドコモの HP から提供されている i アプリ開発ツールである iαppli Development Kit [9] に付属するエミュレータ上で実行した様子を示す．図 5 中の"SE:"は開始タグ（startElement），"At:"は属性（Attribute），"EE:"は終了タグ（endElement），"C:"は文字データ（Characters）の各イベントを意味している．つまり，

図 5 実行の様子
Fig. 5 The situation of execution.

図 5 の内容は図 4 の XML 文書の 8 行目の内容を表示している．図 5 では，エミュレータ上での実行の様子を示したが，同様のサンプル AP を F503i 上で実行し，同様の実行結果を得ている．

この XML 文書の内容を表示するサンプル AP とは別に，XML 文書の解析時間を測定する AP も作成した．つまり，コールバック用のメソッドは何も処理を行わない．この AP を F503i 上で実行させ，図 4 に示した XML 文書を読ませたところ，解析時間は 1 秒程度であった．解析した XML 文書が小さかったこともあるが，低処理速度である F503i 上での実行で，不快感を感じない程度で解析処理を終えることができた．また，同様の AP を F503i 上で実行させて，様々な大きさの XML 文書を読み込ませてみた．読み込ませた XML 文書はすべてデータ指向型の XML 文書で，文書指向型の XML 文書に比べマークアップの密度（タグの出現率）は高いものである．その結果，約 1 KB 程度の XML 文書の解析時間は 3 秒程度，約 2 KB 程度の XML 文書の解析時間は 6 秒程度，約 4 KB 程度の XML 文書の解析時間は 12 秒程度であり，XML 文書の大きさと解析時間の長さが比例していることが確認できた．

この実行結果では，1 KB 程度の XML 文書を 1 秒以下で処理するという目標値には及ばなかった．しかし，この結果をふまえると，504i 以降のシリーズでは今回使用した F503i より処理速度が高速であり，多少大きな XML 文書であってもユーザに不快感を与えることなく解析処理を行えるのではないかと考えられる．

また，DTD もサポートしている TinyXML や Minimizer（Min が読み込みこめる形に変換するプログラム）が付加している Min に比べ，cSAX は非常に軽量である（**表 6**）．そのため，今回示したサンプル AP のように 503i 上でも動作させることが可能であった．

表 6 各パーサと仕様の比較
Table 6 The comparison of parsers and specifications.

		cSAX	TinyXML	Min
仕様	対象 XML	compact XML(*1)	XML	Minimal XML(*1)
	要素	○	○	○
	XML 宣言 処理命令 コメント	○	○	×
	属性 空要素 混在内容	○	○	×
	CDATA 定義済み実体	○	○	×
	DTD	×	(*2)	×
	実体参照			
API		compact SAX(*3)	独自仕様	SAX1.0 (*3)
サイズ		6 KB	23 KB	28 KB

*1 XML のサブセット．
*2 DTD による妥当性検証は行わない．
*3 SAX1.0 のサブセット．

さらに，cSAX は軽量でありながら，Min に比べ多くの仕様をサポートしており，フルセットの XML 文書との互換性が高い．また，TinyXML は DTD をサポートしているが，妥当性検証を行わないため，DTD の機能としては不十分である．そして，cSAX では標準的な仕様である SAX をサブセット化した compact-SAX を API として採用しているため，SAX のスキルによりプログラミングすることが可能であった．実行速度については，TinyXML や Min は J2ME 上で動作させることができないことから比較評価を行っていない．

8.2 FML

FML はブロック管理をリンクリスト型で実現している．前述したように FML は FileManager，File，FileInputStream，FileOutputStream，FileWriter，FileReader の 6 つのクラスから構成されている．**表 7** に FML 内の 6 つのクラス容量について示してある．実際に利用する場合は表中の右側にある JAR ファイル化されたクラス容量となる．この表からも分かるように，FML 全体の JAR ファイルのサイズは約 8.0 KB 程度となる．cSAX 同様，FML も目標値とした 10 KB 以下に抑えられており，この点から FML についても十分に省メモリ化していると考えられる．

実行速度についても，cSAX 同様にターゲットとしている 504i 以降のシリーズでの実機による動作確認は行っていないが，F503i を用いて代表的なメソッド

表 7 FML 内の各クラス容量
Table 7 Class capacity about FML.

クラス名	容量（.class）	容量（.jar）
FileManager	約 3.2 KB	約 1.6 KB
File	約 2.9 KB	約 1.5 KB
FileOutputStream	約 3.2 KB	約 1.8 KB
FileInputStream	約 2.0 KB	約 1.1 KB
FileWriter	約 1.7 KB	約 1.0 KB
FileReader	約 1.6 KB	約 1.0 KB

表 8 FML のベンチマーク
Table 8 The benchmark of FML.

クラス. メソッド	ms	説明
FileManager.create	460	FileManager の作成. ※測定時はサイズ 5 KB で作成した.
FileManager.open	80	FileManager を開く.
FileManager.close	210	FileManager を閉じる.
FileOutputStream. コンストラクタ	3620	ファイル入出力用のバイナリ出力 ストリームを作成.
FileOutputStream. write	60	データの書き込み. ※測定時は 1 KB のデータを一括 して書き込んだ.
FileOutputStream. close	370	ストリームを閉じる.
FileInputStream. コンストラクタ	180	ファイル入出力用のバイナリ入力 ストリームを作成.
FileInputStream. read	50	データを読み込む. ※測定時には 1 KB のデータを一 括した読み込んだ.
FileInputStream. close	10	ストリームを閉じる.

について簡単なベンチマークを実行した（**表 8**）.

ただ，この際に用いた F503i は，503i シリーズにおける Java プログラムの実行速度が一番遅い．これは，簡単なベンチマークプログラムにより，実際に確認している．504i 以降のシリーズでは当然その実行速度も向上しており，504i 以降のシリーズでのベンチマークではもっと良い結果を得られるだろう．各メソッドの実行時間を相対的に比較してみると，FileOutputStream のコンストラクタ生成時間が飛びぬけて実行時間を要する処理であることが分かる．この原因として考えられるのが，リンクリスト型のブロック管理によるランダムアクセスの多発である．Scratchpad に対するランダムアクセスはオーバヘッドの多い処理となる．FileOutputStream のコンストラクタ生成時に，このランダムアクセスが多発するため，このような測定結果となったと考えられる．しかし，これはリンクリスト型からマッピングテーブル型の管理方式に変更することで対処できる．これによって，ランダムアクセスを抑えることが可能であり，これは今後の課題となる．

また，表 7 に示した FileManager クラス以外のファイル入出力関連メソッドの測定結果と FML を用いないで直接 Scratchpad へアクセスした場合と比較してみた．その結果，FML を介した場合は，介さない場合に比べて全体的に 100 ms 以内ではあるが，処理時間を要している．しかし，これもマッピングテーブル型の管理方式にすることで，ある程度は遅延を抑えられる．504i 以降のシリーズでは，今回使用した F503i より処理速度が高速であるため，マッピングテーブルを用いることで，処理速度に関する問題も解決できると考えている．

FML と cSAX の両ライブラリを合わせたミドルウェア全体でも，その容量は 14 KB 程度である．ミドルウェア容量の目標値としてあげた 20 KB よりも 5 KB 程度小さく抑えられており，ミドルウェア全体においても省メモリ化されているといえるだろう．さらに，今後の携帯電話用 Java 実行環境の大容量化もにらめば問題ない結果といえるだろう．また，この結果により，XML を携帯電話用 Java 実行環境向けにサブセット化した compactXML の有効性が示せたと考えている．

9. まとめと今後の課題

本稿では，クライアント指向型のフレームワークを提供するために，i アプリ上での XML の管理・解析を実現するミドルウェアについて述べた．そして，XML を携帯電話用 Java 実行環境に適合させるためにサブセット化した compactXML，その compactXML を解析するための API として SAX1.0 をサブセット化した compactSAX を策定し，これらを利用することで本ミドルウェアではメモリやパフォーマンスに対する負荷を抑えている．本ミドルウェアを利用することで，プログラマは i アプリ上で XML 文書を利用することが可能になるとともに，わずらわしいファイル情報の管理から解放される．これにより，より簡単に Web 上の XML を利用したコンテンツと連携した実用的な AP の開発が可能となる．

また，今後の課題としては，以下の 3 点を考えている．

(1) cSAX の改良
前述したように，現版の cSAX では CDATA セクションや定義済み実体については未対応であり，早急にこれらに対応させる必要がある．

(2) 実用規模のアプリケーションプログラムの開発による評価
現時点では，本ミドルウェアを利用した実用的なサンプル AP が存在しない．そこで，本ミ

ドルウェアの有効性を示すためにも，サーバと効果的に連携した実用性のある AP を開発を通じて，仕様およびミドルウェアの実用規模での評価を行う必要がある．

(3) FML の改良

前述したが，現在の FML はリンクリスト型を用いて実装しており，実行速度が遅いという問題がある．そこで，マッピングテーブルを用いて実装し直す必要がある．

現在は，cSAX の改良を行っており，その後にはターゲットデバイス上での評価を行うとともに，サンプル AP を作成することで，本ミドルウェアの有用性を示したいと考えている．

参 考 文 献

1) i モード対応 Java コンテンツ開発ガイド（詳細編）．http://www.nttdocomo.co.jp/p_s/imode/java/pdf/jguide010514.pdf
2) SAX The Simple API for XML. http://www.saxproject.org/
3) Taivalsaari, A., Bush, B. and Simon, D.: The Spotless System : Implementing a Java System for the Palm Connected Organizer. http://research.sun.com/techrep/1999/smli_tr-99-73.pdf
4) Extensible Markup Language (XML) 1.0, 2nd Edition. http://www.w3.org/TR/2000/REC-xml-20001006
5) SyncML. http://www.openmobilealliance.org/syncml/
6) Crimson. http://xml.apache.org/crimson/
7) Xerces2 Java Parser Readme. http://xml.apache.org/xerces2-j/index.html
8) Min — Minimal XML Parser. http://www.docuverse.com/min/
9) TinyXML. http://www.kvmworld.com/articles/techtalk/
10) iαppli Development Kit. http://www.nttdocomo.co.jp/p_s/
11) Mahmoud, Q.H.: *Learning Wireless Java*, O'REILLY (2002).
12) XML Schema. http://www.w3.org/TR/xmlschema-2/
13) Relax. http://www.xml.gr.jp/relax/
14) 益子由裕，並木美太郎：携帯電話向けの XML 処理用ミドルウェアの開発，マルチメディア，分散，協調とモバイル（DICOMO 2003）シンポジウム論文集，pp.765–768 (2003).
15) 益子由裕，並木美太郎：携帯電話用 XML 処理系のライブラリ開発，情報科学技術フォーラム FIT（2002）講演論文集，M-69, pp.171–172 (2002).
16) 新井イスマイル，和泉順子，中村　豊，藤川和利，砂原秀樹：携帯情報端末における XML 代理サーバによる属性情報管理機構の提案，マルチメディア，分散，協調とモバイル（DICOMO 2003）シンポジウム論文集，pp.769–772 (2003).
17) 轟木伸俊，北村操代：J2ME によるシンクライアントの実現，情報科学技術フォーラム FIT（2002）講演論文集，M-66, pp.165–166 (2002).

(平成 15 年 7 月 31 日受付)
(平成 15 年 11 月 11 日採録)

益子　由裕（学生会員）

1979 年生．2002 年東京農工大学工学部情報コミュニケーション工学科卒業．同年，同大学大学院工学研究科情報コミュニケーション工学専攻修士課程入学．携帯電話やユビキタスネットワークに興味を持ち，モバイルアプリケーションに関する研究に従事．

並木美太郎（正会員）

1984 年東京農工大学工学部数理情報工学科卒業．1986 年同大学大学院修士課程修了．同年 4 月（株）日立製作所基礎研究所入社．1988 年東京農工大学工学部数理情報工学科助手．1989 年電子情報工学科助手．1993 年 11 月電子情報工学科助教授．1998 年 4 月情報コミュニケーション工学科助教授．博士（工学）．オペレーティングシステム，プログラミング言語，ウィンドウシステム等のシステムソフトウェア，並列処理，コンピュータネットワークおよび日本語情報処理の研究・開発に従事．ACM，IEEE 各会員．