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

2007－MBL－40 (28)
2007－UBI－13 (28)
2007／2／23

カスタマイズ可能な携帯端末向けユーザインタフェースの実装と評価

佐藤 充子 岡田 英悟 中本 幸一
兵庫県立大学大学院 応用情報科学研究科
〒650-0044 兵庫県神戸市中央区東川崎町 1-3-3
E-mail: aa05i206@ai.u-hyogo.ac.jp, nakamoto@ai.u-hyogo.ac.jp

ハイパーリンクを用いたカスタマイズ可能な携帯端末向けユーザインタフェースについて示す．携帯端末の利用機会は増えている．そのため，情報端末は使いやすく便利なものであることが望まれる．しかし，現在の携帯端末のインタフェースは複雑なものもある．そして，そのインタフェースは機種によっても異なる．数ある情報端末の中でも、我々が最もよく利用する端末として、携帯電話に焦点をあてる．W3C の XLink と WAP の EFI の機能を利用したメニュー構造について述べる．利用者によるユーザインタフェースのカスタマイズを実現するツールとその実行環境を提案する．そして，この提案したフレームワークの実装と評価を行った．

Implementation and Evaluation a Customizable User Interface Framework using Hyperlinks for Smart Devices

Mitsuko Sato, Eigo Okada and Yukikazu Nakamoto
Graduate School of Applied Informatics, University of Hyogo
1-3-3, Higashi-kawasaki-cho, Chuou-ku, Kobe 650-0044, Japan

A new customizable interface for mobile devices based on hyperlink associability is presented. The opportunity of using mobile devices is increasing. Then mobile devices should be easy to use as a convenience. However many current devices have complex and widely varying interfaces.We focus herein on the user interface of a phone and present the customizable menu structure of the phone using XLink defined in W3C and the External Function Interface (EFI) defined in the WAP Forum specifications. To implement the proposed framework, we have developed a design tool to customize the user interface with hyperlinks and a runtime environment, which manages the objects generated by the tool with the hyperlinks, to evaluate the framework.

1 はじめに

日々の生活において，情報端末を使用する機会が増えている．数ある情報端末の中でも，われわれが最もよく利用する端末として，携帯電話がある．通話の機能しか持っていなかった携帯電話であるが，近年の情報技術の発展により，現在では，テレビ電話，ワンセグ放送の受信，おサイフケータイなどに代表される様々な機能が搭載されている．その機能を実際に利用するためには，数多く用意された機能の中から自分が求めるものを探し出さなければならない．機能は，メニュー構造として保持されている．このメニュー構造がユーザインタフェースの要となる．ユーザインタフェースとして画面に表示される操作盤はこのメニュー構造を基に生成される．そして，利用者がその検索を容易にするため，メニュー構造内では機能をその目的や働きごとにまとめ利用者を目的の機能へと誘導する．しかし，現在の携帯電話では，機能の数が多すぎるため，自分の求める機能がある場所の特定に時間を要する．日常生活においてよく利用するものであればあるほど，より快適に利用したいと考える．その快適さを提供するものの1つとして，使いやすさが挙げられる．様々な観点から使いやすさを捉えることができるが，ここではメニュー構造とその操作方法によりもたらされるものに焦点をあてる．

まず，ユーザインタフェースの要となるメニュー構造についてであるが，提供されるメニュー構造は固定されたものであることが多い．機能の利用には，そのメニュー構造の中から自分が求める機能を探し

出す必要がある．求める機能がメニュー構造の始めにあれば，そのメニュー構造はその利用者にとって使いやすいものといえる．しかし，求める機能は人により異なるものである．たしかに，万人がよく使うと予想される機能は，使用開始後すぐに見つけることができる場所に配置されている．だが，自分が求める機能がメニュー構造の奥に配置されている場合も考えられる．その場合，このメニュー構造はその利用者にとって非常に使いにくいものとなる．さらに，その機能を頻繁に利用する利用者であった場合，その利用は快適とは言い難いものとなってしまう．

この不便さを回避し，より快適に利用できる携帯端末の機能が必要である．このための１つの解決策として，メニュー構造を利用者自身が好みに合わせて作ることが考えられる．現在提供されている携帯電話によってはメニュー構造を一部カスタマイズ可能なものもあるが，カスタマイズできる範囲には制限がある．よって，現在使用中のメニュー構造に関係なく，その端末を利用する利用者がもっとも快適に利用できるように，利用者自身がメニュー構造を作り，そして，利用できる携帯端末の機能が必要である．

次に，利用する端末に関係なく，同じ操作方法で同じ機能を呼び出せる携帯端末の機能が必要となる．機能を呼び出すための操作性について，初めこそ，その複雑さに困惑したとしても，順応していくことができる．そして，一度順応できれば，たとえその操作が若干複雑なものであったとしても，同じ操作を行うことに不便さを感じなくなるかもしれない．しかし，現在提供されている携帯電話の種類は多く，その端末ごとに操作方法が異なることが少なくない．利用する端末が変わることにより，習熟した操作環境から離れなければならないことも考えられる．たとえ，利用する端末が変わったとしても，使い慣れた操作環境をそのまま利用し続けることができれば，この問題を回避することができると考える．

近年の携帯電話は，音声での通話のほかに動画を伴った通話，インターネットへの接続，スケジュールや個人情報の管理などのさまざまな機能を持つものが多い．そして，この多くの機能の中から，利用する機能を効率よく利用できるように工夫されている．現在提供されているものでは，よく利用する機能をある場所にひとつにまとめたり，移動させたりすることによりメニューをカスタマイズできるものがある．例として，本研究でターゲットマシンとして使用するノキア 6680[1] と同じ MIDP を採用している東芝製 910T[2] とシャープ製 911SH[3] を挙げる．910T には，メインメニュー・サブメニュー項目の並び替え，サブメニュー履歴表示，メニュー画像の変更，フォーカス色・文字色の変更の機能がある．911SH には，画面のトータルコーディネイト，シンプルメニュー表示，待ち受け画面のカスタマイズの機能がある．しかし，ユーザがカスタマイズすることができる部分は，画面に表示される各種アイコンや背景といったグラフィック部分であることが多い．また，ここでのカスタマイズ結果は，その端末でのみ有効であり，メーカの異なる機種への変更を行った場合などには，カスタマイズ結果を持っていくことができない．

こうした製品開発研究として以下のものが挙げられる．

- Netfront Dynamic Menu : Access[4]

  標準的な Web テクノロジーを利用できるユーザインターフェースフレームワークである．また，マークアップ言語によりアプリケーションの制御も行うことができる．

- UIEngine : UIEvolution[5]

  ユーザインタフェース開発には XML を使用する．また，開発側はプラットフォームの違いを意識する必要がない．

- VIVID UI : Arcodea[6]

  利用者は，さまざまな GUI を使用用途や自らの好みなどによって選択，利用できる．また，提供者は，それらの提供を容易に行える．

- uiOne : Qualcomm[7]

  色，フォント，音，機能をカスタマイズ，および，携帯電話の本質的な機能である通話，ネットワーク接続などを直接扱うことができる．

- Automotive UI ToolKit : Microsoft[8]

  XML を利用した開発が可能である車載情報端末用プラットフォームのユーザインタフェース開発ツールである．XML で開発したデータの入れ替えで，GUI の変更が可能である．

筆者らは，上記の問題を解決するために，ハイパーテキストを利用したカスタマイズ可能なユーザインタフェースフレームワーク Hyrax を提案した [9]．ここでは，その実装と評価を行ったので，報告する．

2 機能仕様

携帯電話におけるカスタマイズ可能なメニュー構造例の検証により，カスタマイズ可能な携帯端末向けユーザインタフェースに要求される機能の検証を行う [10]．図 1 に，メニューの例を挙げる．

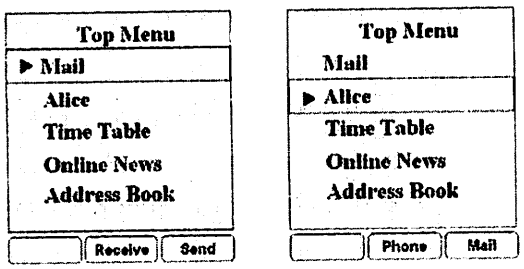

図 1: 携帯電話のメニュー構成例

図 1(a) は，ある利用者がよく使うものを集めて作ったトップメニューである．図 1(a) には，5 つの項目（Mail, Alice, Time Table, Online News, Address Book）が表示されている．

図 1(a) では，最初の項目である Mail にフォーカスがあたっている．図 1(a) 下部のプログラマブルキー表示領域を確認すると 2 つの機能が割り当てられていることがわかる．メールプログラムの主要な機能であるメールの送信と受信の機能が，右ソフトキーと選択キーそれぞれ割り当てられている．これにより，ユーザはトップメニューにある Mail 項目から 1 回のクリックによって 2 つの行動を開始することが可能となる．

図 1 の項目として Alice という人物名が登場する．これは，利用者が頻繁にアリスと連絡を取るためである．項目として ' Alice ' を作成することにより，トップメニューにおいて Alice に対する行動を指定できる．図 1(b) は，Alice にフォーカスがあたったものである．図 1(b) 下部のプログラマブルキー表示領域を確認すると Phone と Mail の 2 つの機能が割り当てられていることがわかる．これらは，Alice への通話と Alice へのメール送信機能であることを意味している．Alice との主な連絡手段を割り当てることにより，Alice との連絡手段の呼び出し手順が簡略化される．その他にもよく利用する機能へとつながる項目を作成することにより，それらの機能の呼び出しや情報へのアクセスが容易になる．

上記のようなメニュー構造やユーザインタフェースには以下に記述する条件が必要である．

- 柔軟なメニュー項目の構成が可能であること．

  メニュー上で，利用者が選択するものは，機能項目と情報項目の 2 種類である．機能項目の選択により，該当する機能が呼び出され，情報項目の選択により実際のデータ，または，そのデータが保存されているディレクトリのいずれかの指定が行われる．トップメニューには，情報オブジェクトが 5 つ（Mail, Alice, Time Table, Online News, Address Book）ある．そして，プログラマブルキーに割り当てられた動作はオブジェクト構造内のオブジェクト間リンクとして説明することができる．この柔軟なメニュー項目の構成を実現するために，Hyrax において，メニュー間のリンク，および，機能の選択はハイパーリンクによって記述する．

- 単一メニューアイテムから複数の操作を単一クリックで可能であること．

  1 つの項目から，複数の行動を指定したい場合が考えられる．図 1(a) の場合，Mail, Alice という 1 つの項目から 2 つの行動を選択実行できることが必要である．そして，1 つの項目から 2 つの行動を選択実行できることにより，より少ない操作での利用開始が可能となる．

- コンテキスト依存メニューであること．

  コンテキストに依存したカスタマイズができるべきである．求められる機能が選択する場所によって異なることから，コンテキストに依存したメニューが必要である．

- 利用時に変更可能なメニュー構成であること．

  携帯電話利用中にもカスタマイズできることが望ましい．不便な箇所やよりよい方法を発見したとき，その場所で，メニューの表示順を変

更や表示名の変更といった作業を行うことができれば，より使いやすいと考えるからである．

機能オブジェクトは，呼び出そうとしている機能と一致したプログラム実体である．オブジェクトとメソッドによって実装される．情報オブジェクトはメニュー項目と他の情報オブジェクト，または，機能オブジェクトへのリンクを含むものである．

図 2 に図 1 で示したユーザインタフェースのメニューを実現するオブジェクト構造を示す．

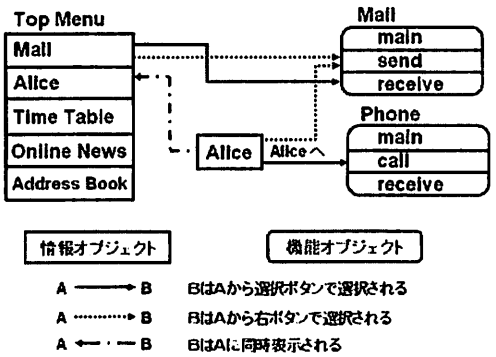

図 2: 図 1 のオブジェクト構造

メニュー項目とその項目から起動する機能を結びつける定義には，EFI(External Function Interface) を利用する．EFI[15] は OMA で規定され，携帯端末の外部機能を呼び出すスキーマを定義している．EFI の XML での表記方法は以下のとおりである．

```
efi: "//" Server "/" [Service]
                    ["?" Parameters]
```

Server 部分で，サービスを提供する EFI アーキテクチャのコンポーネントを識別する．Service 部分で Server によって提供されたサービスを識別する．

Hyrax 実行環境はターゲットマシンの上でのユーザインタフェースプログラム実行環境とフレームワークを提供する．ターゲットマシンにおけるユーザインタフェース機能をブラウザで実現する．このブラウザは，XML 形式で記述されたファイルを読み込み，その記述に従ってオブジェクト構造を構築し，ユーザインタフェースとして提供する．この XML 形式で記述されたファイルを，以降オブジェクト記述ファイルと呼ぶ．図 3 に Hyrax の実行環境を示す．

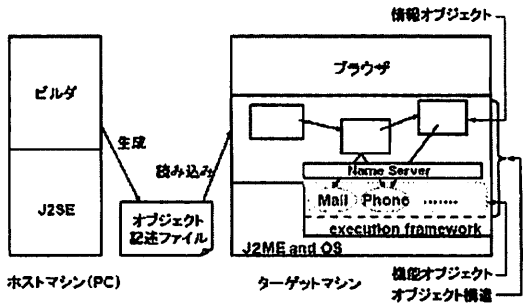

図 3: ビルダ，および，Hyrax 実行環境構造

ハイパーリンク構造を直接記述して，メニューを作成することは困難である．その理由として以下のことが挙げられる．

- 直感的な作業が困難である．ハイパーリンク構造の記述は，テキスト形式で行うため，利用者は作成したいメニューをイメージしながら，そのイメージをテキストに変換する作業を行わなければならない．
- ハイパーリンク構造の記述について事前に定められたルールを把握していなければならない．
- 手入力で作成するため，入力ミスが発生する可能性がある．
- 作業途中で，作成中のメニューがイメージどおりになっているか確認する方法がない．

これらの問題を回避し，メニューの設計を簡単に行えるユーザインタフェースデザインツールが必要である．ユーザインタフェースデザインツールを利用することによって，記述ルールを意識しながら，メニューを作るという煩わしさがなくなる．このユーザインタフェースデザインツールを，以降ビルダと呼ぶ．

- ビルダ

  ブラウザで使用するオブジェクト記述ファイルの生成を行う．ビルダでは，ブラウザでの表示イメージを確認しながら，メニューの作成を行うことができる．そして，メニュー作成中では，表示イメージを直接操作して，表示順や表示名の変更，項目の削除などの操作を行うことができる．

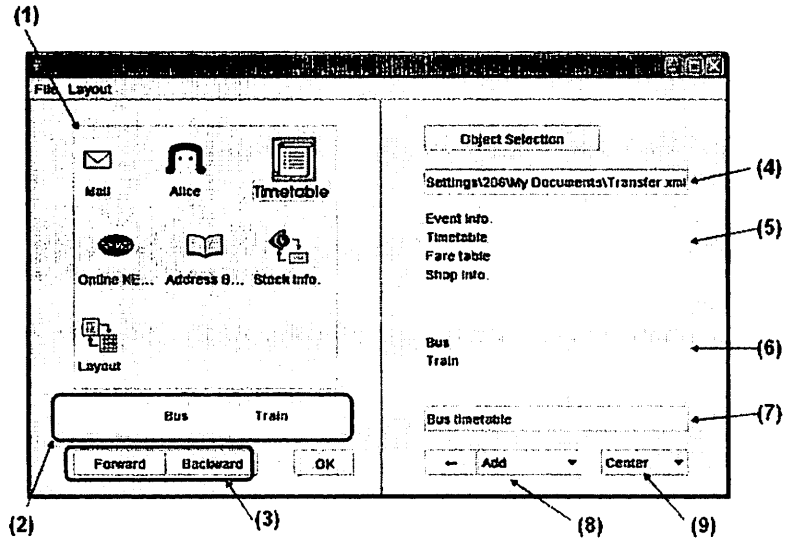

図 4: ビルダのメイン画面

- ブラウザ

  ビルダによって生成されたオブジェクト記述ファイルの解析を行い，その解析結果からオブジェクト構造を構築する機能をもつ．そして，構築されたオブジェクト構造をユーザインタフェースとして提供する．また，機能呼び出しやリンク先への移動といったオブジェクトの操作機能を持つ．

3 ビルダ

このビルダには，以下の機能が必要である．

- ビルダ上での出力イメージの表示

  ビルダでの作業中に，ブラウザでの表示を確認できることにより，作成中のものがイメージどおりになっているかの確認ができる．

- 出力イメージの編集機能

  出力イメージ上で，対象となる項目の編集を行えることにより，直感的な作業が可能となる．

- 出力イメージのハイパーリンク構造記述への変換と出力

  ビルダで作成したイメージをブラウザで利用できるハイパーリンク構造記述形式へと変換し，出力する．これにより，記述形式のルールを意識することなくファイルの作成が可能である．また，ファイルを自動生成するため，入力ミスによるエラーを防ぐこともできる．

ユーザインタフェースの設計を容易にするビルダの詳細について記述する．

図 4 にこのビルダのメイン画面を示す．画面構成は以下のとおりである．

- 出力イメージ表示部 (1)

  携帯電話でのメニュー表示イメージの表示を行う．表示できるレイアウトは，リスト表示タイプとアイコン表示タイプの 2 種類である．これらのレイアウトは，イメージ作成中に切り替えることも可能である．図 4 は，アイコン表示タイプのものである．

- プログラマブルキー設定部 (2)

  携帯電話でプログラマブルキーに割り当てられた処理を知るためのメッセージを表示する．

- イメージ表示順変更ボタン (3)

  出力イメージ表示部に表示されている項目の順番を変更する際に使用する．

- オブジェクト選択部 (4)

情報オブジェクト，機能オブジェクトの記述がなされたファイルの選択を行う．

- 項目選択部 (5)

  オブジェクト選択部で選んだオブジェクト記述ファイルには複数の項目が含まれている可能性がある．オブジェクト記述ファイルに記述されている項目をリスト形式で表示する．

- リンク先選択部 (6)

  項目選択部で選ばれた項目のリンク先の指定が表示される．

  リンク先には，情報オブジェクトにリンク指定記述された項目と機能オブジェクトに含まれるメソッド名がある．

- パラメータ部 (7)

  メソッド起動時にパラメータを渡す必要がある場合，そのパラメータを記述する．

- 追加ボタン (8)，(9)

  項目選択部，または，リンク先選択部で選択した項目を出力イメージに追加する．デフォルトでは，選択キーに起動処理の追加を行うが，任意で他のプログラマブルキーに追加することも可能である．また，パラメータが設定されていた場合，そのパラメータも共に追加される．

4 実装と評価

4.1 実装

ビルダの実装には，J2SE 上に swing を使用して構築した．そして，ブラウザの実行時環境の実装には，Sun Microsystems 社提供の J2ME Wireless Toolkit の PC エミュレータとターゲットマシンとして携帯電話であるノキア 6680 を使用した．使用言語は，Java プログラミング言語である．Java 実行環境は組み込み機器向けの Java 言語使用である「J2ME」の一部として定義されている CLDC 1.0[11] と MIDP2.0[12] である．パーサは kXML2(バージョン 2.2.2)[13] に基づく．

図 5 にターゲットマシンの Hyrax 実行環境上で実行した写真を示す．

図 6 に，Hyrax 実行環境のコードサイズの一覧を示す．kXML2 を改造した XML パーサのサイズ

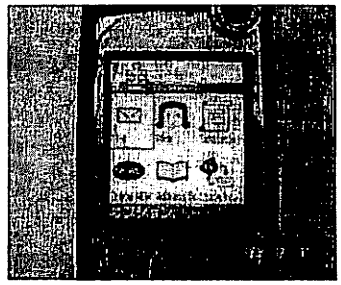

図 5: 携帯電話における Hyrax ブラウザの写真

が相対的に大きいが，ターゲットマシン上での実行においては十分小さいため，Hyrax 実行環境の構成は可能である．

クラス名	行数
ブラウザ部分	1,514
Name Server	59
XML パーサ	6,717
File Server	60
合計	8,350

図 6: コードサイズ

4.2 評価

ユーザインタフェースとしてメニューの表示を行う Hyrax ブラウザの性能評価を行うために，メニュー画面の呼び出しにかかる時間の測定を行った．

また，Hyrax では，XML 形式のオブジェクト記述ファイルに保存されたオブジェクト構造を元にインタフェースの作成を行っている．そこで，プログラム内にオブジェクト構造が直接記述されたものとの比較を行った．評価のための時間測定箇所は，以下のとおりである．

まず，ブラウザ起動後からトップページの表示が完了するまでに必要な時間を測定した．Hyrax ブラウザが，起動時に行う処理の概要は以下のとおりである．

1. 外部記憶メモリに保存されたオブジェクト記述ファイルのメモリへの取り込み．

2. 画面表示用フォームの作成．

3. オブジェクト記述ファイルからメニュー項目の抽出，および，フォームへの登録．(コンテキスト依存メニュー表示の設定含む．)

4. 表示用フォーム完成後，表示面に設定．

現段階では,Java プログラムがファイルにアクセスするたびにユーザ許可が必要である．この許可発行による不便さを回避するために，ブラウザ起動時にオブジェクト記述ファイルをメモリへ保存している．しかし，Hyrax ブラウザが Java アプリケーションとして信頼されたものとなれば，このファイルアクセスへの制限が緩和される．そこで，今回の時間計測ポイントを画面表示用フォームの作成から表示面に設定完了までとする．テストケースとして，表示項目数が異なる 3 種類を用意した．用意したメニューの表示項目数は，それぞれ 4，8，12 である．表示時間計測はそれぞれ 20 回ずつ行い，その平均値をとる．

図 7 に，計測結果を示す．

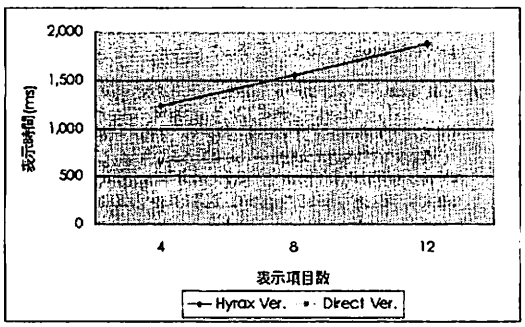

図 7: 初期画面表示時間計測結果

Hyrax ブラウザ，直接記述タイプともに，表示項目数の増加により表示にかかる時間は増加する．しかし，その増加幅は，Hyrax ブラウザの方が大きくなっている．これは，Hyrax ブラウザが行う直接記述タイプと異なる点であるオブジェクト記述ファイルの構文解析，要素の抽出などを行っていることが要因と考えられる．

つぎに，トップメニューから次階層のメニューが表示されるまでの時間を測定した．Hyrax ブラウザが，表示面をトップメニューから次階層のメニューに変更する処理の概要は以下のとおりである．

1. トップメニューで行われた選択の判別．(選択先が情報オブジェクトであることを確認する．)

2. オブジェクト記述ファイルからメニュー項目の抽出，および，フォームへの登録．(コンテキスト依存メニュー表示の設定含む．)

3. 表示用フォーム完成後，表示面に設定．

時間計測ポイントは，トップメニューで選択が行われてから表示面に設定完了までとする．テストケースとして，表示項目数が異なる 3 種類を用意した．用意したメニューの表示項目数は，それぞれ 4，8，12 である．表示時間計測はそれぞれ 20 回ずつ行い，その平均値をとる．

図 8 に，計測結果を示す．

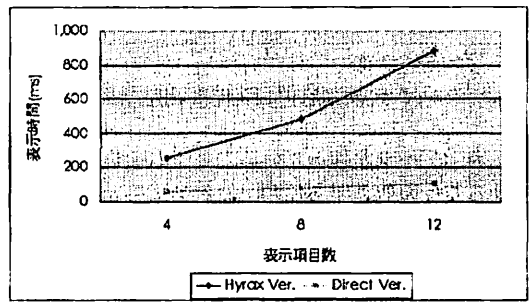

図 8: 次階層初回表示時間測定結果

Hyrax ブラウザ，直接記述タイプともに，表示項目数の増加により表示にかかる時間は増加するが，その増加幅は，Hyrax ブラウザの方が大きくなっている．初期画面表示時間測定結果（図 7）と類似している．

最後に，トップメニューから次階層のメニューが表示（再表示）されるまでの時間を測定した．Hyrax ブラウザが，表示面をトップメニューから次階層のメニューに変更（再表示）する処理の概要は以下のとおりである．

1. トップメニューで行われた選択の判別．(選択先が表示データの残っている情報オブジェクトであることを確認する．)

2. 表示データが残っているフォームを表示面に設定．

時間計測ポイントは，トップメニューで選択が行われてから表示面に設定完了までとする．テストケースとして，使用するオブジェクト記述ファイルは，図 8 の計測を行ったものと同じものを使用す

る．表示項目数は，それぞれ 4，8，12 である．表示時間計測はそれぞれ 20 回ずつ行い，その平均値をとる．

図 9 に，計測結果を示す．

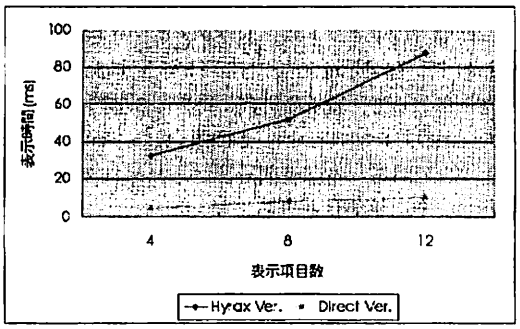

図 9: 次階層再表示時間測定結果

表示データが残っているフォームを再び表示させるだけの処理であるが，これまでの結果と同様に，表示項目数の増加による所要時間の増加幅は，Hyrax ブラウザの方が大きくなっている．

今回の時間測定結果から，オブジェクト記述ファイルの扱い方が Hyrax ブラウザでのメニュー表示所要時間を左右することが判明した．オブジェクト記述ファイルサイズの増大にともなって，処理に要する時間が増加する．だが，表示しようとする項目数が今回調査した範囲内であれば，いずれの値においても使用に耐えうるものであると考える．

5 まとめ

Hyrax はハイパーリンクを用いた携帯端末向けのカスタマイズ可能なユーザインタフェースフレームワークである．Hyrax によって実現されたユーザインタフェースはユーザの好みにあわせてカスタマイズ可能である．ハイパーリンクを用いて，携帯端末のユーザインタフェースをカスタマイズできるツール，および，そのツールによりハイパーリンクを用いて生成されたオブジェクトの管理を行う Hyrax 実行環境の開発を行い，このフレームワークの評価を行った．W3C の XLink[14] と WAP における EFI[15] の機能を利用することにより，ユーザインタフェースのカスタマイズが可能になる．

参考文献

[1] ノキア, Series60 Developer Platform 2.0: 仕様 *Original Document: "Series 60 Developer Platform 2.0: Specification"*, バージョン 1.0, 2004 年 1 月.

[2] 株式会社 東芝, SoftBank 910T, http://www.toshiba.co.jp/product/etsg/cmt/softbank/910t/910t_menu.htm, 2006.

[3] シャープ株式会社, SoftBank 911SH, http://www.sharp.co.jp/products/sb911sh/index.html, 2007.

[4] Access, Netfront Dynamic Menu, http://www.jp.access-company.com/products/nf_mobile/dynamicmenu/index.html, 2007.

[5] ITX イー・グローバレッジ株式会社, UIEngineTM の技術解説書, http://e-globaledge.com/mobile/uiengine.pdf, 2002 年 9 月.

[6] Acrodea, VIVID UI, http://www.acrodea.co.jp/ir/business.html, 2006.

[7] Qualcomm, uiOne, http://brew.qualcomm.com/brew/ja/about/uione.html, 2007.

[8] Microsoft, Microsoft Releases Windows Automotive 5.0 for Accelerating Customized In-Vehicle System Development, July 12, 2005.

[9] Yukikazu Nakamoto and Mitsuko Sato, "Design of A Hyperlink-based Application Framework for Smart Devices", Proceeding of The 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing(ISORC 2006), IEEE Press, pp.261-268, April 2006.

[10] 佐藤充子, 中本幸一, "ハイパーリンクを用いた携帯端末向けユーザインタフェースの一実現方式", 第 50 回システム制御情報学会研究発表講演会 8F4-4, 2006 年 5 月.

[11] Sun Microsystems. Connected Limited Device Configuration, Specification version 1.0,Java 2 Micro Edition,May 2000.

[12] JSR118 Expert Group, Mobile Information Device Profile for Java 2 Micro Edition, Version 2, Nov. 2002.

[13] Stefan Haustein, kXML, http://kxml.sourceforge.net.

[14] W3C, *XML Linking Language (XLink) Version 1.0*, June 2001.

[15] Open Mobile Alliance.*External Functionality Interface Framework, Candidate Version 1.1 9-Jun-2004*, 2004.