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

2007－DBS－143（1）
2007／7／2

組込みシステム向けHMIツールのモデルベース開発

新 吉高† 鯨井 俊宏†† 土井 敬司†† 深谷 直彦††
原 隆浩††† 西尾章治郎†††

† エイチ・シー・エックス 〒140-0002 東京都品川区東品川4-12-6 日立ソフトタワーB 14階
†† 日立製作所 〒185-8601 東京都国分寺市東恋ヶ窪1-280
††† 大阪大学 〒565-0871 大阪府吹田市山田丘1-5
E-mail: †yoshitaka.atarashi.uw@hitachi.com ,
††{toshihiro.kujirai.ek,keiji.doi.zm,naohiko.fukaya.ha}@hitachi.com, †††{hara,nishio}@ist.osaka-u.ac.jp

あらまし 組込みシステムは製品の多機能化により，HMI開発工数が増大している．市販ツールによる設計は個々の製品の特性に合わせ，使い方に制約を持たせる必要があったため，製品系列や世代を超えたHMI設計情報の共有が難しかった．そこで，本稿では組込みシステム向けHMIツールのモデルベース開発について議論する．開発したツールでは，市販ツールからの変換規則を定め，データベースでモデルを統合管理することで，市販ツールの使い方に依存しない設計情報の管理を可能とした．

キーワード 組込みシステム，HMI，モデルベース開発，XML, RelaxNG

Model-based Development of a HMI Tool for Embedded Systems

Yoshitaka ATARASHI†, Toshihiro KUJIRAI††, Keiji DOI††, Naohiko FUKAYA††,
Takahiro HARA†††, and Shojiro NISHIO†††

† HCX Corporation Higashi Shinagawa 4-12-6, Shinagawa-ku, Tokyo, 140-0002 Japan
†† Hitachi Ltd. 1-280 Higashi-Koigakubo, Kokubunji-shi, Tokyo, 185-8601 Japan
††† Osaka University Yamadaoka 1-5, Suita, 565-0871 Japan
E-mail: †yoshitaka.atarashi.uw@hitachi.com ,
††{toshihiro.kujirai.ek,keiji.doi.zm,naohiko.fukaya.ha}@hitachi.com, †††{hara,nishio}@ist.osaka-u.ac.jp

Abstract Man-month of HMI development for embedded systems is increasing rapidly because a wide variety of functions are required. While it is easy to design a HMI specification using standard office applications, it becomes difficult to share HMI specifications among the product models and generations. In this paper, we propose a model-based development method of a HMI tool for embedded systems. The proposed tool can transform HMI specifications designed by standard office applications into XML-based models which do not depend on how to use the applications.

Key words Embedded System, HMI, Model-based Development, XML, RelaxNG

1. まえがき

組込みシステムはハードウェアスペックの制約を意識しながらソフトウェア開発を行うことが特徴である．自動車エンジン制御に代表される，制御系組込みシステムは制御ロジックのモデル化と，モデルからの自動ソースコード生成により，上流設計への人員シフトを進めたことでソフトウェア開発効率の向上が進んでいる．また，携帯電話，カーナビに代表される情報系組込みシステムは，本来の必要な機能だけでなく，地上波デジタル放送受信機能や音楽プレイヤー機能などを取り込み，多機能化が進んでいる．

多機能化による開発工数の増加を解決する技術としてSPL(Software Product Line) と呼ばれるソフトウェア共通化技術の研究が盛んになっている [1]．しかし，製品毎の特徴が大きく反映されるHMI(Human Machine Interface) の開発は，標準的なモデリングツールが存在しておらず，SPLによる製品系列を越えたソフトウェア共通化の恩恵を得られないため，開発効率向上の決め手は今のところない．

組込みシステムのHMI開発は，PCと比較して画面が狭いという制約を考慮しつつ，多様な情報提供，組込みシステム特

有の特殊なハードウェアデバイスとの接続に対応する必要がある．例えば，一般的なWindowsシステムでは，それぞれのアプリケーションに独立したウィンドウが与えられているため，アプリケーションの画面もそれぞれ独立している．一方カーナビを例に取ると，地図表示・誘導といった基本的な機能以外に，音楽再生や交通情報など様々なアプリケーションが実行される．ユーザにとって見やすい情報提供を行うためにはレイアウトに工夫が必要で，各アプリケーションが独立して画面を設計できないという課題がある．携帯電話も近年の解像度向上により，同時に複数アプリケーションの情報を提供するようになり，カーナビと同様の問題が発生しつつある．

また，PCでは，マウスやキーボードのようなHMIデバイスは標準化されており，標準のI/O APIや，標準のデバイスドライバも提供されている．アプリケーションはこのような標準デバイスを前提として作成すれば良い場合が多い．しかし，組込みシステムは標準と呼べるHMIデバイスは存在せず，各社がより安全で使いやすいHMIを実現するために改良を続けている．アプリケーションのHMIも，多様な専用デバイスの存在を反映した設計となるため，デバイスの差異をHMI設計モデルに反映させる必要がある．

そこで本稿では，PCと異なる特徴を持つ組込みシステム開発に適したHMI開発のために，様々なツールの設計情報を柔軟に扱えるHMIツールのモデルベース開発について議論する．開発したツールでは，市販ツールからの変換規則を定め，データベースでモデルを統合管理することで，市販ツールの使い方に依存しない設計情報の管理を可能とする．

モデルベース開発の有効性と課題を明らかにするため，カーナビのHMI設計情報を用いたプロトタイプによる評価を実施した．

2. 関連研究

2.1 モデルベース開発

モデルベース開発手法としては，OMG(Object Management Group)によって提唱されたMDA(Model Driven Architecture)[2]が有名である．MDAは厳密な設計情報のモデル化による開発効率向上を目的とし，モデルベース設計を行う上で必要となる言語，交換形式を定義したアーキテクチャである．MDAの一実現例であるEMF(Eclipseモデリングフレームワーク)[3]は，メタモデルECoreを定義し，これを中心としてJava，XML，UMLを統合して扱うことが可能なツール開発フレームワークを提供する．EMFは開発途上であるものの，非常に強力なフレームワークであり，将来はモデルベース開発を行う上で必須になると期待される．しかし，Eclipse以外のツールとの連携を実現するには敷居が高いのが難点である．

これに対し，メタモデルの表現をXMLスキーマで定義するXML中心の設計手法がある．この手法を用いた場合，モデルの操作をXMLバインディング，すなわちXMLスキーマから自動生成されるプログラム言語で行うことで，ツールの開発工数を削減できる．XMLは木構造で表現可能なモデルやドキュメントを扱うシステムとの親和性は高いが，EMFやUMLと比較して，双方向参照や参照先の型を定義できないため，表現能力は低くなる．

2.2 モデルベースHMI開発

HMI開発に関しては，XMLによるモデル化のアプローチを取る研究がいくつか行われている．

文献[4]はモデルベースHMI開発の事例である．HMI開発における自動車メーカとサプライヤの関係を示し，従来紙ベースで仕様を取り交わしていたHMI設計から，モデルベース開発によりHMIシミュレーションモデルを互いにブラッシュアップすることで開発プロセスが改善すると主張している．一方，本研究では，要求仕様が必ずしもモデル化されている必要はなく，様々な要求に対応できるようモデルを自由にカスタマイズすることで，多様な要求仕様のモデル化を実現するアプローチを取る．

文献[5]は車載機器における，XMLによるHMI設計情報の標準化に関して報告している．これは車載マルチメディアインタフェースの標準化を目標とするAMI-C(Automotive Multimedia Interface Collaboration)の成果である．Java VMとOSGi(Open Services Gateway Initiative)を前提ソフトウェアとして要求するため，HMI開発効率の向上には寄与しても，組込みシステムとしてのコスト削減要求には応えられない．一方，本研究では，このような組込みシステム上での前提ソフトウェアは要求せず，モデルから必要な情報を抽出し，組込みシステムの制約に合わせた変換ツールを個別開発するアプローチを取る．

2.3 HMIツール

HMI開発のためのツールは市販製品，フリーソフトウェアなどの形態で，既に多数存在する．ここではツールを市販汎用ツール，市販専用ツール，内作ツールに分類し比較する．

（1） 市販汎用ツールの例であるAbode社のPhotoshop[6]，Microsoft社のVisio[7]は公開されたデータフォーマットを採用し，スクリプトによる機能追加を可能としている．様々な目的で利用されるために多くの機能が提供されているが，実際の開発においては，製品の特徴にあわせて使い方に制限を加える等の工夫が必要である．

（2） 市販専用ツールの中には，モデルベースHMI開発を支援するため，独自のデザインツールとCASEツールを提供し，設計情報からシミュレーション実行，さらには実行形式への変換を行うことができるものがある．例としてはILC社のGENWARE2[8]，エレクトロビット社のtresos GUIDE[9]などがある．しかし，このようなツールの導入費用は市販汎用ツールと比較して非常に高く，さらにツールの操作性が独特で習熟に時間がかかることから，大規模な開発になるほど採用が難しい傾向がある．また，採用されたとしても同時利用可能なライセンス数に制約がある場合も多く，開発が佳境になるとライセンスが不足するというケースが散見される．

（3） 内作ツールは開発費用が莫大になる上，使いづらいものになりがちであった．しかし，統合開発環境Eclipse[10]の登場により，本来必要な機能のみEclipseプラグインとして開発ができるようになり，内作ツールでありながらツールの習熟

が容易でかつ大規模な開発に採用しやすい状況が生まれている．

3. 開発方針

3.1 開発目的

本研究は大規模な組込みシステムの HMI 開発を支援するためのツールの提案を目的とする．ここで大規模とは，要求仕様策定者が機能毎に分かれているため，要求仕様のレベルでの調整が難しい状態を想定している．このような場合，機能毎に成熟度が異なるため，成熟度の高い機能ほど要求仕様が簡素に，成熟度の低い機能ほど複雑になる傾向がある．従って，要求仕様の作成方法は統一されておらず，紙ベースか，Microsoft 社の Excel のような市販汎用ツールで主に作成される．部分的には市販専用ツールによる要求仕様モデルの提供の可能性がある．

組込みシステムはハードウェア制約を考慮する必要があるため，要求仕様と実装仕様の間には大きなギャップが存在する．このギャップを埋めるために，要求仕様策定者と開発者の間には，仕様調整担当者が必要となる．

モデルベース開発が実現されていない環境では，要求仕様策定者と仕様調整者の数がほぼ同等で仕様調整が進められる．しかし，要求仕様から実装仕様への落とし込みが開発者の負担になるため，開発者の数は相対的に多くなる．その結果，仕様変更時に影響を受ける開発者を発見するのが難しくなり，作業見積もりが困難なために開発遅延を引き起こす．

モデルベース開発の実現には，仕様調整者によってモデルの詳細化を実施する必要があるため，熟練開発者を仕様調整者に移す必要がある．また，設計されたモデルを統合的に管理するデータベースを持つことで，仕様調整の結果が共有され，仕様変更の影響範囲を追従可能となる．本研究の成果によりこれが実現されれば，上流工程から下流工程に速やかに HMI 設計情報が流れるようになり，全体としての開発効率の向上が期待できる．

3.2 プラットフォーム非依存

組込みシステムは標準化よりも差別化，低コスト化を求める傾向がある．同じ型番の製品でさえ，低コスト化のためにハードウェア設計を変更することもある．従って，OS やハードウェアに依存しないモデル化が必要である．組込みシステムの開発は複数の業者が関わることが多いため，特定のツールやプラットフォームへの依存は，予想外に製品全体の開発費を押し上げる可能性がある．

市販ツールで作成された設計情報を製品特有のモデルに変換するためには，モデルそのものがプログラム言語に依存しないことも重要である．市販ツールは独自スクリプトを採用していることが多いため，実用性の高い Java ベースの技術をモデルの表現手段として採用することは難しい．

3.3 組込みシステムの進化への追従

組込みシステムはバージョンアップや派生製品の開発において，既開発部分の再利用を行っているため，モデルベース設計の導入のためには，既開発資産のモデル化のコストを計算に入れる必要がある．さらに，今後のバージョンアップや派生についても追従可能なように，拡張性が求められる．このような

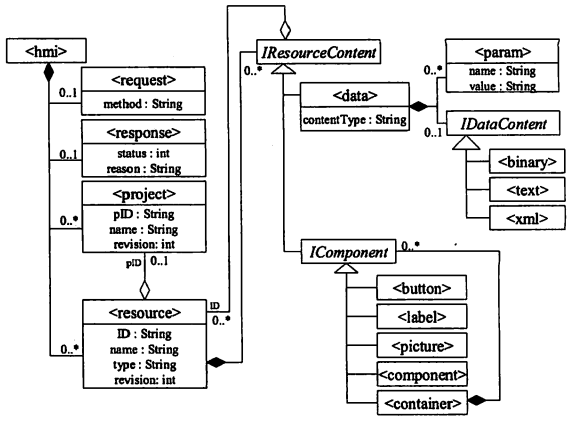

図 1　モデルのクラス図表現

バージョンアップや派生は複数の開発プロジェクトを立てて並行に開発されることが多いため，複数のプロジェクトを跨るモデルの並行利用が可能でなければならない．

4. 提案ツール

本章では提案ツールの中心となるモデルの設計と，モデルベースによる提案ツールの構成について述べる．

4.1 モデルの設計

前章の想定に基づく HMI 設計情報のモデルを RelaxNG [11] を用いて設計した．図 1 はモデルのクラス図表現の主要な部分を抜粋したものである．

hmi タグはルートタグである．HMI 設計情報の一部または全部を表現することができる．

request タグは後述のリソース管理 DB に対する要求を表現するためのもので，最大 1 つ存在できる．method 属性は操作の種別である．リソース管理系の操作としては，getList(リソース一覧の取得)，get(リソースの取得)，insert(リソースの新規登録)，update(リソースの更新)，delete(リソースの削除) を用意している．他にも，リビジョン管理，検索に関するメソッドがある．

response タグは要求に対する返信を表現するためのもので，最大 1 つ存在できる．status 属性は要求の成否を判断するためのもので，HTTP [12] の status と同等のコードを採用した．

project タグは HMI 設計情報を纏める単位であるプロジェクトを表現する．pID 属性を主キーとしてリソース管理 DB で管理される．name 属性はプロジェクト名，revision 属性はプロジェクトの変更に応じて増加するリビジョン情報を表す．

resource タグは HMI 設計情報の管理単位であるリソースを表現する．ID 属性を主キーとしてリソース管理 DB で管理される．name 属性はリソース名，type 属性はリソースの種別，revision 属性はリビジョン情報を表す．また，リソースを管理するプロジェクトの pID を参照として持つ．

IResourceContent はリソースの中身を抽象化したインタフェースを表し，データとコンポーネントから構成される．resource タグへの 0 個以上の参照は，リソースから他のリソー

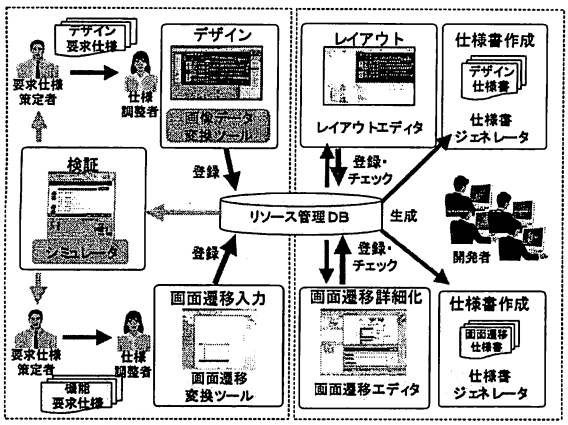

図 2　提案ツールの構成

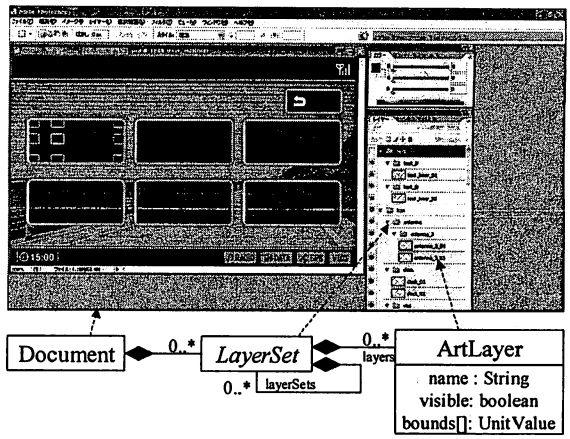

図 3　Photoshop 画像データのモデル

スへの参照が含まれる可能性があることを示す．

data タグはデータを表し，データの中身そのものと，データの性質を表す情報から構成される．IDataContent インタフェースで表されるデータの中身は，文字列，フォント，色，画像，音声等，単体で意味を持つ情報が対象である．テキスト形式，バイナリ形式 (XML 内部では BASE64 形式で表現)，XML 形式のいずれかで保持する．param タグはデータの性質を現すための情報で，データ 1 つに対し複数持つことができる．例えば，スキン，言語，昼または夜，製品の納入先などの違いで画像だけが入れ替わる，というような場合に用いる．

IComponent インタフェースは，button タグ，label タグ，picture タグで表現される基本 GUI(Graphical User Interface) 部品，component タグで表現される対象ハードウェアの CPU，OS，プログラム言語等に依存する実装部品，container タグで表現される部品の集合を表すコンポーネントの抽象化表現である．

画面レイアウトは container によるツリー構造で表現する．また，画面遷移のうち状態遷移に関する部分については SCXML [13] をデータとして持つことで表現する．

4.2　提案ツールの構成

図 2 に，前節のモデルベースの提案ツールの構成を示す．HMI 設計情報を統合管理するリソース管理 DB は，Servlet をインタフェースとし，HTTP/POST で XML を送受信する通信プロトコルにより，モデルの統合管理を可能とする．

画像データ変換ツールは画像データからモデルへの変換機能を持つ．スクリプト言語 [14] によりレイヤー単位で画像を抽出し，リソース管理 DB に登録する．画面遷移変換ツールは画面遷移データからモデルへの変換機能を持つ．Visio は XML 形式で内部情報を出力可能であり，これを解釈して画面遷移情報に変換し，リソース管理 DB に登録する．モデルへの変換さえできれば，どのようなツールからの変換ツールでも作ることは可能である．

レイアウトエディタはリソース管理 DB と連携し，レイアウト編集，コンポーネント定義によるモデル操作が可能な Eclipse プラグインである．画面遷移エディタはリソース管理 DB と連携し，画面遷移編集によるモデル操作が可能な Eclipse プラグインである．シミュレータとの連携により，画面遷移のシミュレーションが可能である．

仕様書ジェネレータはリソース管理 DB からモデルを抽出し，PDF 形式または HTML 形式の仕様書を生成可能なアプリケーションである．どのツールで作られた設計情報かという区別なしに仕様書を自由に生成できるため，提案ツールの構成に対して新たなツールを追加しても，モデルの設計が変わらない限りは本ツールを変更する必要はない．

これらのツールを使うことで，要求仕様から開発に至るまでの仕様の詳細化にかかる時間が短縮できると期待される．特に，シミュレータにより，段階的に画面デザイン仕様と画面遷移仕様の確認が可能となり，要求仕様策定者と仕様調整者の間での仕様詳細化が協調的に行えるようになる．

5.　HMI 設計情報のモデル化

本章では，提案ツールを用いた HMI 設計情報のモデル化手法について述べる．

5.1　画像データ変換

図 3 は，Photoshop 付属マニュアルの JavaScript Scripting Reference [14] を参考に，Photoshop 画像データのモデルの一部をクラス図表現したものである．Document はドキュメント全体，LayerSet はレイヤーの集合，ArtLayer は画像を管理する単体のレイヤーを現す．ArtLayer の name 属性はレイヤー名，visible はレイヤーの表示状態，bounds は画像が定義されている範囲を表している．

画像データ変換の方法は，レイヤー名の命名規則を定め，4.1 節で定義したモデルに変換することである．ここでは，レイヤー名として，「コンポーネント名__ボタン状態__昼夜状態__スキン名」のように__（半角のアンダーバー）をデリミタとする命名規則を定義する．

まず，モデルのコンポーネントへの変換を行う．レイヤー名でボタン状態が省略された場合は picture に変換する．ボタン状態が定義された場合は button に変換する．コンポーネント

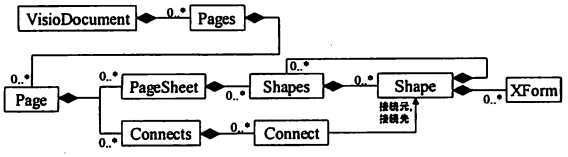

図 4　Visio データのモデル

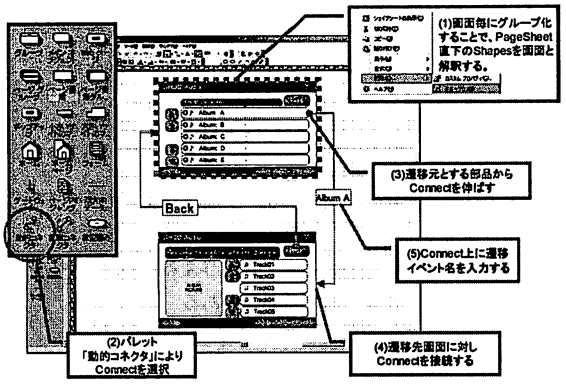

図 5　Visio による画面遷移設計

名は resource タグの name 属性にマッピングする．bounds は PhotoShop の管理する単位 (UnitValue) であるため，ピクセル単位の値に変換し，component の属性として原点からの相対座標と幅，高さにマッピングする．

次に，モデルのデータへの変換を行う．レイヤーから bounds の範囲内の画像を PNG 形式で切り出し，data タグの下の binary タグにマッピングする．レイヤー名ボタン状態，昼夜状態，スキン名はそれぞれ，data タグの下の param タグにマッピングする．コンポーネントに対してはこの data タグへの参照を持つよう設定する．

以上の変換を行う際に，画像データ変換ツール内でレイヤー名命名規則とモデルへの変換表を定義し，それに従って変換することで，Photoshop の仕様に依存しないモデル化が実現できる．一般的には，定義すべき部品の情報は製品種類や世代，派生によって異なることが多いため，この変換表そのものをプロジェクト単位で管理する方法が有効である．

5.2　画面遷移変換

Visio2003 以降のバージョンでは，図 4 に示すようなクラス構成を持つ XML ファイルを生成可能となっている．VisioDocument タグはドキュメント全体，Pages タグは複数ページ，Page タグは単一ページを表す．

Page タグは主に，形状を持つオブジェクトを管理する PageSheet タグと，接続を管理する Connects タグから構成される．Shapes タグはグループ化操作によって纏められる形状の集合を表す．Shape タグは形状を表し，Shapes タグへの参照を持つか形状情報を含む XForm タグを持つ．Connect タグは Shape タグへの参照を持つことで接続元と接続先を表す．

Visio を用いて画面遷移をモデルに変換するために，図 5 に示すような手順を実施する．

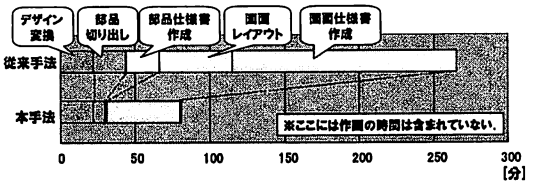

図 6　画面デザイン仕様策定の工数削減効果見積もり

（1）　画面のグループ化により PageSheet タグ直下の Shapes のみを画面遷移の画面とみなせるようになる．モデルへの変換は，Shapes を container にマッピングする．

（2）　画面遷移定義のため，動的コネクタを選択する．

（3）　遷移元の画面から動的コネクタを伸ばすことで，Connect タグの接続元情報が定義できる．

（4）　遷移先の部品に動的コネクタを伸ばすことで，Connect タグの接続元情報が定義できる．モデルへの変換は，SCXML の transition タグに対して，(1) で定義した container の ID をマッピングする．

（5）　動的コネクタに名前を設定する．SCXML の transition タグのイベント名にマッピングする．

Visio データに含まれる画像情報とレイアウト情報は XForm タグから抽出できる．Shapes は container，Shape は picture としてマッピングできる．また，Connect の接続元に当たる Shape を button としてマッピングすることで，Visio データだけで簡単な画面遷移シミュレーションを行えるようになる．

6.　有効性評価

6.1　画面デザイン仕様策定

画面デザイン仕様策定における提案ツールの工数削減効果を示す．測定は 2 名で行い，1 画面当たりの平均時間を計った．

（1）　デザイン変換: 要求仕様策定者が市販汎用ツールで作成したデザイン要求仕様を仕様調整者が受け取る．組込みシステムの場合色数に制限あることが多いため，デザイン要求仕様からの減色，色補正をここで行う．提案ツールではこの作業工数は削減できない．測定の結果 22 分であった．

（2）　部品切り出し: 画像データ変換ツールによって画像部品への変換を行い，リソース管理 DB に登録することで，提案ツールによって作業が自動化される分工数が削減する．測定の結果 8 分であった．

（3）　部品仕様書作成: 仕様書ジェネレータにより部品仕様書を自動生成することで，提案ツールによって作業が自動化される．測定の結果 1 分であった．

（4）　画面レイアウト: 登録された画像部品を元に，レイアウトエディタを用いて必要な画面数分のレイアウトを行い，リソース管理 DB に登録する．提案ツールにおいても，この作業工数は削減できない．測定の結果 49 分であった．

（5）　画面仕様書作成: 仕様書ジェネレータにより画面仕様書を自動生成することで，提案ツールによって作業が自動化される．測定の結果 1 分であった．

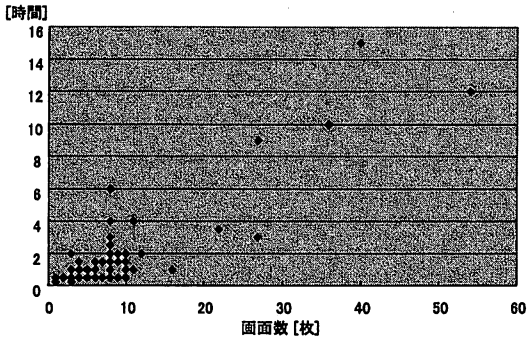

図 7　画面遷移入力時間

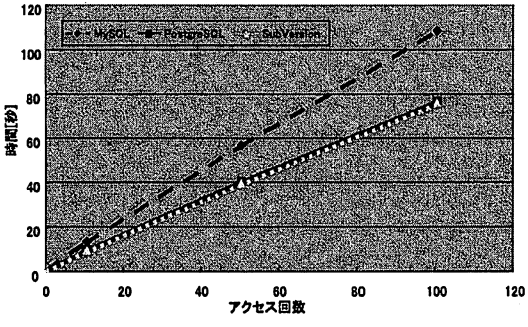

図 8　リソース管理 DB 性能

比較対象として 2005 年の開発事例を調査した．290 枚の画面デザイン仕様書作成について部品仕様書と画面仕様書を市販のドローツールで作成しており，1 人 1 枚当たりの平均時間は，(1)(2)(3) を合わせて 66 分，(4)(5) を合わせて 199 分であった．

これらを纏めると図 6 のようになる．要求仕様の画面仕様書化にかかる工数が 7 割近く削減できることから，仕様調整の TAT(Turn Around Time) の短縮の見通しが得られた．

6.2　画面遷移仕様策定

Excel や紙ベースでの画面遷移仕様を元に，全体の画面遷移の整合を取るのは非常に大変なため，モデル化された仕様をシミュレータによって検証することで工数削減が可能となる．しかし，モデル化そのものの作業に時間がかかることから，まず Excel ファイルで作成された画面遷移仕様 100 枚に対して，提案ツールを用いてモデル化にかかる時間を測定した．

図 7 は横軸に 1 枚の画面遷移仕様に含まれる画面数，縦軸にモデル化にかかった時間をプロットした散布図である．画面 1 枚当たり平均 12 分の作業時間で画面遷移仕様のモデル化が実行できているが，画面数が多くなるにつれて接続の複雑さが増すため，作業時間は画面数に比例せず，2 乗に近い伸びを示す．

以上から，画面数の少ない画面遷移仕様については，仕様調整の TAT の短縮の見通しがある．しかし，組込みシステムの複雑な画面遷移仕様を記述する場合に，接続線を結ぶだけの画面遷移設計手法では難しいことが分かった．状態遷移表による設計など，より厳密で網羅的な定義が可能な設計ツールを導入する必要がある．

6.3　リソース管理 DB のアクセス性能

リソース管理 DB のバックエンドとして，JDBC 経由でアクセスする MySQL 5.0.21，PostgreSQL 8.2 と，WebDAV 経由でアクセスする Subversion 1.3.0 を用いたアクセス性能評価を行った．評価に用いた PC のスペックは WindowsXP SP2, Pentium4-3GHz，RAM 1GB である．

484 バイトの XML データを登録し取得する操作を繰り返し，得られた結果が図 8 である．PostgreSQL と Subversion はほとんど性能差が無く，MySQL が少し遅い．しかし 1 アクセスに 1 秒弱かかり，全体的に遅いのは XML 形式のモデル操作を多用していることが一因と推測している．

モデルベース開発の効果により，多様な DB を切替えて使うことが容易だった反面，DB のアクセス性能の遅さはツールを使った製品開発効率そのものに影響を与えるため，ボトルネックを調査し性能改善を実施する必要がある．

7. むすび

本稿では，組込みシステムの特徴を踏まえ，モデルベースによる HMI ツールの開発手法を提案した．実際の組込みシステムの HMI 設計情報をプロトタイプ適用し，モデル化を通じて開発工数削減の見積もりと，モデル化工数そのものの負荷に関する見通しが得られた．また，多様な要求仕様に対応するために，市販ツールの活用だけではなく，モデルベース設計による内作ツール開発の有効性が見えてきた．

今後は，製品開発全体を見据えた工数削減を目指し，モデルの進化によるツールへの影響の解析，開発フェーズにおける設計情報の検索性能を考慮したリソース管理 DB の評価，HMI 設計情報と実装ソースコードの間のギャップの解析を行う予定である．

文　献

[1] 吉村健太郎: 製品間を横断したソフトウェア共通化技術 -ソフトウェアプロダクトラインの最新動向-, 情報処理学会会誌, Vol.48 No.2 pp.171-176 (2007).
[2] The Model-Driven Architecture, http://www.omg.org/mda/.
[3] Frank Budinsky et al. : Eclipse Modeling Framework: A Developer's Guide, Prentice Hall Ptr (2003).
[4] Carsten Bock: Model-Driven HMI Development: Can Meta-CASE Tools do the Job?, Proc. of HICSS (2007).
[5] Laci Jallis, Frank Szczublewski: AMI-C Content-Based Human Machine Interface (HMI), Proc. of SAE (2004).
[6] Photoshop CS2, http://www.adobe.com/jp/products/photosh
[7] Visio 2003, http://office.microsoft.com/ja-jp/visio/default.aspx
[8] GenWare2., http://www.ilc.co.jp/Products/genware2.htm.
[9] tresosGUIDE, http://www.elektrobit.jp/3soft/product-info.ht
[10] Eclipse, http://www.eclipse.org/.
[11] J Clarck, M Murata: RELAX NG Specification, http://www. relaxng. org/spec-20011203.html (2001).
[12] RFC2068 Hypertext Transfer Protocol – HTTP/1.1, ftp://ftp.isi.edu/in-notes/rfc2068.txt.
[13] SCXML, http://www.w3.org/TR/scxml/.
[14] JavaScript Scripting Reference, Photoshop CS2/スクリプティングガイド/JavaScript Reference Guide.pdf.