Vol.2011-SE-174 No.7
2011/11/2

実行可能な画面プロトタイプによる業務システム開発手法の提案

野尻周平[†1]　橋本康範[†1]　三部良太[†1]
石川貞裕[†1]　山口潔[†2]

近年，業務システムの画面向けに動的な振る舞いを備えた高機能ユーザインタフェース (UI) を採用する事例が増えてきている．高機能 UI の開発においては実行可能プロトタイプを用いた確実な要求獲得が必要不可欠であるが，初期開発工数が追加で必要となるという課題がある．そこで本論文では，実行可能な画面プロトタイプ向けの軽量開発手法を提案する．提案手法の特徴は，異なる実行環境に対応した機能部品を設けることにより，同一の画面設計情報に基づいて画面プロトタイプおよび本番システムの自動生成を実現することである．自動生成ツールの試作と，テストプロジェクトでの実証実験により，提案手法の実現性と有用性を確認した．

A Method to develop Business System Based on Excutable Screens Program Prototype

Shuhei NOJIRI,[†1] Yasunori HASHIMOTO,[†1]
Ryota MIBE,[†1] Sadahiro ISHIKAWA [†1]
and Kiyoshi YAMAGUCHI [†1]

High-performance user interface (UI) enabling dynamic screen behaviors have been widely applied for business systems recently. Developing executable prototype is a key activity to analyze requirements of such UIs with users, however, also needs additional initial costs. In this paper, we propose a novel approach to develop executable prototype for business systems. In our approach, both prototype and deliverable systems are generated from same data specifications of screens with applying functional components for different operating environments. We illustrate the concept with case studies to evaluate the feasibility and benefit of our approach.

1. はじめに

近年のソフトウェア開発において,「開発者がいかにソフトウェアを作るか」だけでなく「利用者にいかにしてソフトウェアを活用させるか」についての議論が活発化するなど[1)2)]，優れたユーザインターフェイス (UI) への取り組みが脚光を浴びている．他方，現在多くの企業は，プログラムの配布や更新が容易である Web アプリケーション (Web アプリ) により業務システムを提供しているが，Web アプリは，クライアントとなるブラウザの表現能力に制約されるため，ユーザ操作性を作りこみにくいという問題もある．こうした背景の中，この Web アプリの問題を解決し，高いユーザ操作性を持つプログラムを容易に配布，変更できるフレームワークとして Rich Internet Appications (RIA) などが登場し，RIA を利用した高機能な UI を採用する事例も増えてきている．

しかし，高機能な UI を業務システムに適用する際は，要求定義段階においていかに画面に対する要求を獲得し，いかにして顧客とベンダの間で認識が不一致の無いように合意するかということが大きな課題となる．画面に対する要求には，画面の静的な表示に対する要求と，画面の振る舞いに対する要求とがある．静的な表示とは，例えば画面上に配置されるオブジェクトのレイアウトや色味といった，時間による変化が原則的に無い画面表示情報のことであり，画面の振る舞いとは，例えば，ユーザが画面上のあるコントロールで一定時間ポインタを滞留させたら，入力を助けるコードヘルパーを表示するといった動的な表示や，画面の切り替えにかかる秒数など，時間やイベント，タイミングに応じて変化する画面表示情報のことである．

高機能な UI 開発においては，このような画面の振る舞いについても要求を確認する必要があるため，プロトタイプの活用が有効である．なぜならば，従来の Web アプリの UI 開発では，ブラウザの表現能力の制約から静的な表示について設計すれば十分であったが，高機能な UI 開発する際は，静的な表示の設計のみでは不十分であり，利用者のアクションに応じた画面の振る舞いについても設計する必要があるためである．静的な表示のみであれば，例えばペーパープロトタイプ[3)] や画面定義書などの図面によって顧客とぶれの無いコミュニケーションを図ることができるが，画面の振る舞いについては図面によるコミュニ

†1 (株) 日立製作所　横浜研究所
Hitachi, Ltd., Yokohama Research Laboratory
†2 (株) 日立製作所　情報・通信システム社
Hitachi, Ltd., Information and Telecommunication Systems Company

ケーションは難しい．

しかしながら，プロトタイプの活用には次の課題がある．

(1) 本番開発[*1]を前提としない画面プロトタイプは，要求の実現性確認が十分にできないため，仕様欠陥による手戻り発生の可能性がある

(2) 本番開発への流用を前提とした画面プロトタイプの作成は工数がかかる

本研究は，この課題を解決するため，本番開発を前提とした画面プロトタイプの開発において，画面設計情報からのデータ構造自動抽出技術に基づくプロトタイプ生成手法を提案する．本手法により，プロトタイプを本番開発へ移行する際に無駄となってしまう開発作業の工数が削減可能となり，画面プロトタイプの迅速性と確実性の両立を実現できる．

2 章で本研究の研究課題について詳しく述べた後，3 章では提案手法について述べる．4 章では提案手法の適用例について述べる．5 章では，提案手法の評価と考察を述べる．6 章では関連研究について述べ，7 章で本研究の成果と今後の展望についてまとめる．

2. 研 究 課 題

本章では，まず本研究が対象とする画面プロトタイピングについての概要と課題について述べた後，本研究が取り組む研究課題について明らかにする．

2.1 画面プロトタイピングの概要

本節では，業務システム開発における画面プロトタイピングの目的とプロセスについて述べる．

2.1.1 画面プロトタイピングの目的

画面プロトタイピングの目的は，画面に対する顧客要求を具体的な試作品により明確化し，画面要求に関する仕様欠陥を削減して後工程からの手戻りを防ぐことである．画面プロトタイプを通して画面要求に関するコミュニケーションを図り，要求の認識不一致による手戻りの防止を図る．

なお，プロトタイピングの対象とする画面は，開発対象となる業務システムにおける主要な画面と，複雑な仕様となることが想定される画面を抽出したものであり，業務システムの全ての画面を対象とするわけではない．

2.1.2 画面プロトタイピング全体のプロセス

図 1 に，企業システム開発における画面プロトタイピングの流れを示す．画面プロトタイピングは，要求定義プロセス[4] の中で実施される．業務分析や既存システム分析の結果に基づき実施し，その結果は本番画面のデザインガイドライン作成や画面仕様の定義に用いられる．

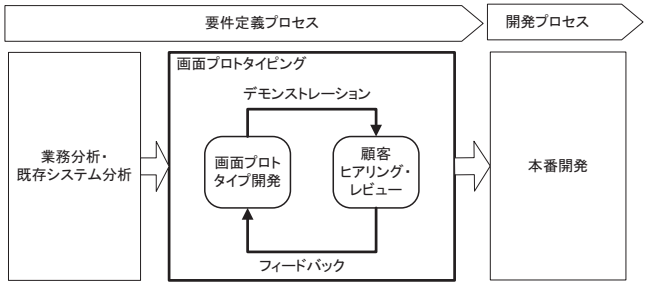

図 1 画面プロトタイピングの流れ

画面プロトタイピングが実施される要求定義プロセスは，一般的には 1 から 2 カ月程度である[5]．また，図 1 中に示した，デモンストレーションとフィードバックの，1 サイクルの期間は数日から 1 週間程度である．

2.1.3 画面プロトタイプ開発のプロセス

次に，開発するプロトタイプの性質の違いに着目し，図 1 の画面プロトタイプ開発のプロセスを分類すると，代表的には以下に示す 2 つのプロセスがある．

模擬型画面プロトタイプ開発

模擬型画面プロトタイプとは，ペーパープロトタイピング[3] に代表されるような，本番ソフトウェアとは独立した言語やフレームワーク，表現方法を用いて模擬的に画面の表示や振る舞い再現したプロトタイプである．図 2 に模擬型画面プロトタイプ開発のプロセスを示す．

模擬型画面プロトタイプは，開発対象を画面表示プログラムに絞り，オーサリングツールや画面プロトタイピング用ツールなどを用いて作成する．

実践型画面プロトタイプ開発

実践型プロトタイプとは，アジャイル開発[6] に代表されるようなプロセスにり，本番ソフトウェアの言語やフレームワークを用いて，実践的に画面の表示や振る舞いを開発するプロトタイプである．実践型画面プロトタイプ開発のプロセスを図 3 に示す．

[*1] 本番開発とは，製品ソフトウェアに搭載するプログラムコードおよび関連する設計成果物を作成することを指す．

実践型画面プロトタイプは，実際に動作させ画面や画面間の連携を確認するため，画面と画面振る舞いに関するプログラムに加えてプロトタイプ対象の画面に関連する機能やデータ保管に関するプログラムについても開発する．

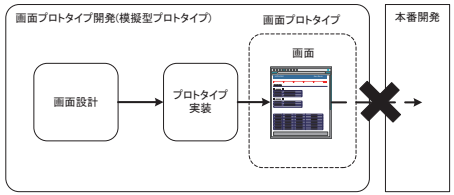

図 2 模擬型画面プロトタイプ開発のプロセス

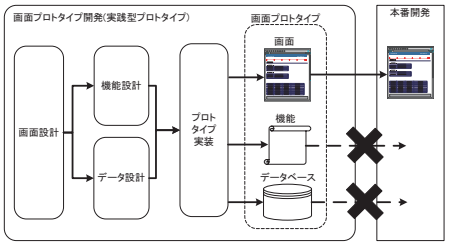

図 3 実践型画面プロトタイプ開発のプロセス

2.2 画面プロトタイピングの課題

高機能 UI を持つ画面開発において，2.1.3 節で述べた画面プロトタイプ開発のプロセスを実施する際の課題について述べる．

2.2.1 模擬型画面プロトタイプ開発プロセスの課題

模擬型画面プロトタイプ開発は，開発対象が限定されているため，短期間で開発できるというメリットがある．一方で，本番ソフトウェアと異なる開発言語・フレームワークにおいて開発されるため，以下に示すデメリットがある．

画面成果物が本番開発に流用できない

開発されたプロトタイプ成果物をそのまま本番ソフトウェアに流用することができないため，プロトタイプとして開発した画面と同仕様の本番ソフトウェア向け画面を再開発する必要がある．

顧客要求の実現性検証が十分に実施できない

模擬型画面プロトタイプを用いて抽出した仕様が，本番ソフトウェアの開発言語やフレームワークにおいては実現不可能な仕様や，実現可能であったとしてもその実装に多大な工数を必要とする仕様である可能性がある．

2.2.2 実践型画面プロトタイプ開発プロセスの課題

実践型画面プロトタイプ開発は，本番ソフトウェアと同一の開発言語・フレームワークにおいて開発されるため，顧客要求の実現性検証を十分実施できる．また成果物となる画面プログラムを本番開発においても活用できる．しかし一方で，以下に示すデメリットがある．

画面プロトタイピングの 1 サイクルに多くの時間を要する

模擬型画面プロトタイプ開発と比較し，画面間を連携させる機能やデータ保管に関するプログラムの開発が必要となるため，画面プロトタイプの開発に時間を要する．

本番ソフトウェアに貢献しない設計を要する

実践型画面プロトタイプにおける機能やデータ保管に関する設計・実装は，あくまで限定されたスコープのプロトタイプを動作させるための簡易的なものである．また，画面プロトタイプと本番ソフトウェアでは，ソフトウェアのアーキテクチャが異なることが一般的であるため，画面プロトタイプの全ての成果物を本番開発へ流用することは難しい．

2.3 本研究における課題設定

本研究の課題を，迅速かつ確実な画面プロトタイピングを実現するため，実践型プロトタイプ開発のロス工数を最小化することとする．

2.3.1 ロス工数の定義

本研究におけるロス工数の定義を以下に示す．

ロ ス 工 数

画面プロトタイプ開発に投入されたが，本番ソフトウェアの開発に直接貢献しない工数を指す．例えば，捨てられる成果物 (プログラム) の実装にかかった工数や，最終製品に反映されなかった仕様を検討した工数のことである．なお，ここでの工数とはプロトタイプ開発に投入された開発者の作業時間である．

2.3.2 課題の分析

表 1 に 2.2 節で述べた模擬型画面プロトタイプと実践型画面プロトタイプのトレードオフをまとめた結果を示す．表中の迅速性とは，画面プロトタイピングのサイクルをいかに短期間で実施することができるかを示し，確実性とは，画面プロトタイピングによって獲得した仕様に欠陥の少なく，手戻りが発生しにくいことをを示す．

表 1 模擬型画面プロトタイプと実践型画面プロトタイプのトレードオフ

	迅速性	確実性
模擬型プロトタイプ	○	×
実践型プロトタイプ	×	○

2.1.1 節でも述べた通り，画面プロトタイピングの目的は手戻りの防止である特に，業務システムの開発では，開発コストにより強く制約される．開発プロセスに入ってからの仕様修正，および仕様修正に伴う手戻りは多大な追加コストを発生させるため，避けることが望ましい．

2.3.3 本研究の課題

本研究の課題は，実践型画面プロトタイプ開発プロセスの課題の迅速性をプロセスのロス工数の削減により改善し，迅速性と確実性を両立する画面プロトタイピングプロセスを確立することである．

3. 画面中心プログラム自動生成ツールによる画面プロトタイピング手法

本章では，画面中心プログラム自動生成ツールによる画面プロトタイピング手法を提案する．画面中心プログラム自動生成ツールは，画面設計情報からデータ構造を自動的に抽出し，このデータ構造に基づいて画面プロトタイプの動作のために補完が必要なプログラムを自動生成する．

3.1 アプローチ

以下に示すアプローチにより，2.3 節で述べた課題の解決を図る．

(1) 実践型画面プロトタイプ開発プロセスを分析しロス工数となる作業を識別する
(2) ロス工数作業の設計情報のうち，中核をなすデータ構造を，画面定義プログラムより自動抽出する
(3) 抽出されたデータ構造に基づき，ロス工数作業の設計情報を補完，生成し実行可能な画面プロトタイプを自動生成する

実践型画面プロトプロトタイプ開発プロセスのロス工数作業をツールによって自動化することにより，画面プロトタイプ開発において本質的に必要な作業が明らかになり，迅速化を図ることができる．また，ロス工数作業で設計，実装される設計情報を完全自動生成によって補う事により実行可能な画面プロトタイプによる実現性検証が実施できる．

3.2 ロス工数作業の識別

実事例を通じ，図 3 に示した実践型画面プロトプロトタイプ開発プロセスの分析を行った．表 2 に，プロセスの設計作業で定義する代表的な設計情報について，その設計情報の定義がロス工数となるか否かという観点で識別し，まとめた結果を示す．表 2 中の「識別結果」の値の意味は次の通りである．

- ○：ロス工数とはならない設計情報
- △：本来後工程で設計すべきであるためロス工数となる可能性がある設計情報
- ×：ロス工数となる設計情報

表 2 ロス工数となる設計作業・設計情報識別結果

設計作業	設計情報	識別結果
画面設計	画面名	○
	静的表示	○
	振る舞い	○
	表示データ	○
	イベント	○
機能設計	機能名	○
	機能種別	○
	関係画面	○
	トリガーイベント	○
	入出力データ	△
	ロジック仕様	△
データ設計	データ名	△
	データ型	△
	データ構造	△
	データ間関係	△
プロトタイプ実装	画面プログラム	○
	機能プログラム	×
	データスキーマ	×

3.3 画面中心プログラム自動生成ツールの機能構成

本節では，画面中心プログラム自動生成ツールの機能構成について述べる．

3.3.1 提案ツールの要求仕様

3.2 節で識別したロス工数を，完全自動によって削減するため，提案ツールには以下の機能が必要である．

- データ設計における設計情報を補完する
- 機能選択において設計を省略したデータ入出力およびロジック仕様について補完する
- プロトタイプ用アーキテクチャと本番開発用アーキテクチャのそれぞれに対して機能プログラムとデータスキーマを自動生成する
- 自動生成されたプロトタイプ用アーキテクチャと本番開発用アーキテクチャソフトウェアについて，画面を通じて見たの振る舞いが見かけ上等価となるように自動生成する

3.3.2 ツールによる完全自動生成に向けたプロセス設計

限られた設計情報から完全自動生成を実現するには，生成のための前提条件を与える必要がある．本研究では，以下に述べる 3 つの条件満たすをプロセスを提案し，画面プロトタイプの完全自動生成を実現する．

画面プログラムに対する制約

表 2 によれば，データ設計作業およびデータスキーマの実装はロス工数となる可能性が高く，画面プロトタイピング時の実施は望ましくない．そのため，画面間のデータ連携に関する画面の振る舞いについてデモンストレーションする際には，データに関する情報を自動生成ツールに与える必要がある．

本提案では，画面プロトタイプにおいてデータ連携を実現したい場合，画面プログラム上に既定の方法でデータ名称を記述するようにプロセスを制約する．この際，名称が同じデータについては，同一のデータとして扱う．図 4 は，画面開発を Flex[7] により実施する際に，画面を記述する MXML[⋆1]におけるデータ名称の埋め込み例である．MXML エレメントの id 属性をデータ名称定義に流用し，動的なデータを表示するコンポーネント関しては，id 属性にデータ名称を定義するようプロセスに制約を与えた (図 4 中の id=”EmployeeName” など).

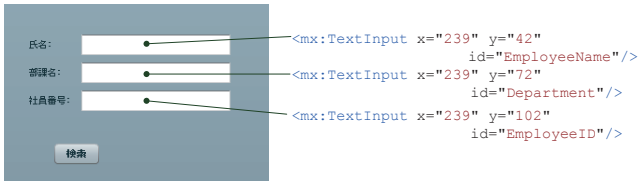

図 4 画面プログラムに対するデータ名称埋め込み例

機能設定に関する制約

画面プロトタイプにおける機能は，プロトタイプを動作させるための補助である．開発者が機能設定において定義する設計情報量を削減するため，選択できる機能種別数を極力絞り込む必要がある．本研究では，業務システムの画面から呼び出される典型的なデータ授受に関する機能を抽出した．表 3 に抽出した機能種別を示す．また，図 5 に機能の入出力に関する模式図をに示す.

⋆1 Macromedia Flex Markup Language

表 3 採用した機能種別

機能種別名	概要	データの入出力
検索	検索条件に適合するレコード群を取得する	画面→画面，データ→画面
レコード選択	選択されたレコード群を取得する	画面→画面
更新	画面上のに表示された値を格納する	画面→データ
ステータス更新	レコードの特定のカラムの値を更新する	画面→画面，画面→データ
追加	レコードを新規に追加して格納する	画面→データ
削除	レコードを削除する	画面→データ

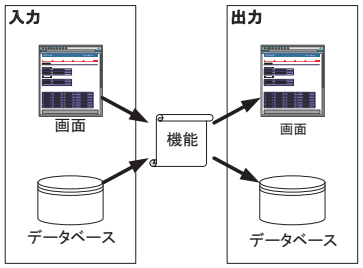

図 5 機能の入出力

ソフトウェアアーキテクチャに対する制約

プロトタイプ用アーキテクチャと本番開発用アーキテクチャに対してそれぞれ自動生成を可能とするため，自動生成ツールが対象とするソフトウェアアーキテクチャを予め定めておく必要がある．

本研究で対象としたアーキテクチャを図 6 に示す.

プロトタイプ用アーキテクチャにおいては，画面プログラム，機能プログラム，データベースがクライアントに全て集約しており，サーバに配置されるプログラムは存在しない．これは，画面プロトタイピングにおいては，サーバの実装は検討の対象外であり，また，デモンストレーションの際にサーバを構築する手間を削減するためである．

本番開発用アーキテクチャにおいては，クライアントに画面プログラムとクライアント側機能プログラム，サーバにサーバ側機能プログラムとデータベースを配置する．クライアン

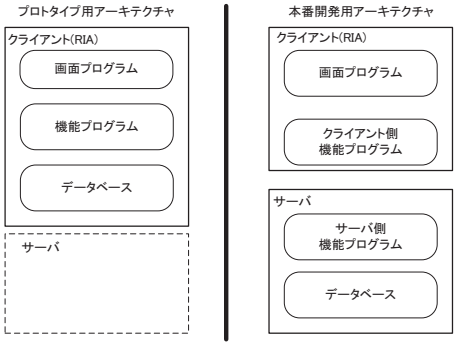

図 6 本研究の対象アーキテクチャ

ト側機能プログラムとは，表 3 において画面間にデータの入出力が閉じる機能のプログラムである．また，サーバ側機能プログラムは，画面とデータベース間で入出力が発生する機能のプログラムである．

3.4 自動生成ツールの機能構成

図 7 に 3.3.1 節で述べた要求仕様，および 3.3.2 節で述べたプロセスの制約に基づく提案手法の機能構成を示す．

図 7 のデータ構造抽出・生成機能が，3.3.2 節で述べた画面プログラムに対する制約と，画面構造，画面に用いられているコントロールの種類に基づき，各画面のデータ構造を抽出する．また，複数画面の分析結果から，同一データの集約およびデータ間の関係を類推し，データ構造を生成する．

機能仕様抽出・補完機能では，3.3.2 節で述べた機能設定に関する制約に従って入力された選択機能について，選択機能に関連する画面と上記で生成されたデータ構造を用いて，機能のデータ仕様を補完し，プログラムの自動生成に向けた機能仕様を生成する．

プログラム自動生成に機能では，入力された実行アーキテクチャと，上記で生成されたデータ構造，機能仕様に基づき，実行アーキテクチャに対応する機能部品を用いて機能プログラム及びデータベーススキーマを自動生成する．生成プログラムのアーキテクチャは，3.3.2 節で述べたソフトウェアアーキテクチャに対する制約に従い，本研究においては図 6 に示したプロトタイプ用，もしくは本番開発用のアーキテクチャとなる．

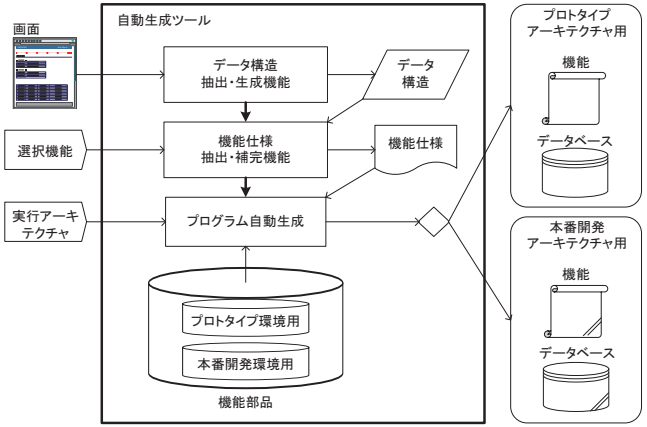

図 7 提案手法の機能構成

3.5 提案手法による画面プロトタイプ開発プロセス

図 8 に，本提案を適用した画面プロトタイプ開発プロセスの概要を示す．これは，表 2 のロス工数となる設計作業・設計情報識別結果に基づき，図 3 に示した実践的画面プロトタイプ開発のロス工数，もしくはロス工数となる可能性があると識別された設計情報の開発作業について，ツールによる自動生成により代替することによって得られるプロセスである．

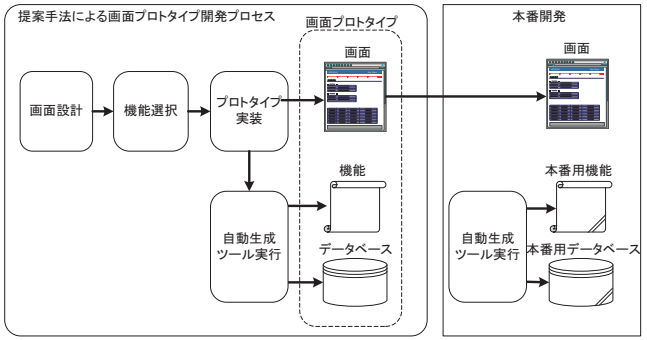

図 8 提案手法による画面プロトタイプ開発プロセス

以下に，図 8 に示した各プロセスに付いて述べる．

画 面 設 計

画面設計では，開発者が従来の画面プロトタイプと同様に画面の静的表示および動的振る舞いについて設計する．

機 能 選 択

機能選択では，開発者が画面プロトタイプによるデモンストレーションを実行するにあたって必要となる画面間の機能を設計する．機能選択による設計は，機能名，機能種別，関連する画面およびトリガーイベントの選択により実施する．

プロトタイプ実装

プロトタイプ実装では，開発者は画面プログラムに関する実装のみ行う．

自動生成ツール実行

自動生成ツール実行では，自動生成ツールが画面プロトタイプ実行に必要な機能プログラムとデータスキーマを生成する．また，本番開発開始時に画面プロトタイプを本番開発へ移行するため，自動生成ツールは本番開発用のアーキテクチャ向けのプログラムを自動生成する．

4. 評価・考察

4.1 実装による実現性評価

図 6 のアーキテクチャにおいて，本提案手法のプロトタイプを実装し，本提案の実現性を評価した．表 4 に本実験で採用したアーキテクチャを示す．

表 4 生成実験対象アーキテクチャ

種別	名称	version
クライアント側実行言語	Flex	3.2
サーバ側実行基盤	Java[8)]	J2EE 1.6
サーバ側データベース	HiRDB[9)]	09-02

画面プロトタイププログラムから抽出した設計情報に基づき，プロトタイピング用アーキテクチャ，本番アーキテクチャへの自動生成が実現できることを確認した．

4.2 試用による有効性評価

画面プロトタイピングの流れを想定した 3 つのテストプロジェクトに対して検証実験を実施した．表 5 にテストプロジェクトの概要を示す．また，表 6 に実験結果を示す．

表 5 テストプロジェクトの概要

	プロジェクトA	プロジェクト B	プロジェクト C
RIA 開発経験	有	無	有
開発画面数	2	3	2
機能数	4	11	4

表 6 テストプロジェクトに対する適用結果

	プロジェクトA	プロジェクト B	プロジェクト C
工期 (学習期間含む)	5 日	5 日	8 日
自動生成機能数	4	11	4

4.3 考　　察

4.3.1 画面プロトタイピングのサイクルのスパン

1 回の画面プロトタイピングのサイクルで開発される 2〜3 画面程度の規模のプロトが，5 日〜8 日で開発できた．本実験では，提案手法のツールの学習期間を含んでおり，実際には 1 日から 2 日程度短縮できると考えられる．1 回の画面プロトタイピングのスパンが数日から 1 週間程度という前提において，本手法は有効に働くことが確認できた．

なお，テストプロジェクト C は，望む画面の振る舞いを実現するため，本手法の提供するフレームワークに対する調査期間が必要だったため，同規模の他のプロジェクトより工数を要した．

4.3.2 工数のロス

工数のロスについては，本番開発を実施する実プロジェクトへの適用ができていないため，定量的な数値は得られていないが，各テストプロジェクトにおいて，本研究がロス工数作業と識別した機能プログラムやデータスキーマに対する編集を行う必要はなかった．このことから，実プロジェクトへの適用実験においても良好な結果を示すことが期待できる．

4.3.3 本手法の制限

本評価では，100 画面程度のシステムを想定し，そのうちプロトタイピング工程を通じて 10 画面程度を試験開発する例を想定して検証した．手法そのものに画面数に制限はないが，プロトタイピング対象となる画面数が増大したり，プロトタイピングに関わる開発者が増大

した場合には，画面に定義するデータ名称の一意性について不整合が生じやすくなるため，自動抽出によるデータ定義の有用性が低くなる可能性がある．

5. 関連研究

UI および RIA のプロトタイピングに関連する研究には，次のものがある．

小形[10] らは，UML 要求モデルから UI プロトタイプを自動生成し，具体的な画面フローを提示することによって要求定義段階における顧客と開発者側の要求認識の齟齬に対する解決を図っている．しかし，本研究は業務の流れであるユースケースの振る舞いを対象としており，また，開発したプロトタイプを本番へ移行することを意図していない．そのため，本研究が扱う RIA などの高機能な UI に対する要求の齟齬の解決には用いることができない．

また，Bozzon[11] らは，モデルベースアプローチとコード自動生成による RIA アプリケーションの開発効率化方法を提案している．しかし，本研究で提案しているモデルは，機能やデータの詳細まで踏み込んだモデル化が必要である．プロトタイプを動作させるため，本来プロトタイピングのフェーズで設計すべきでない仕様についても定義が必要となり，本研究が課題としている手戻り発生要因となってしまう．

6. おわりに

6.1 まとめ

本研究は，高機能な UI を実現する画面プロトタイピングにおいて，獲得要求の実現性確認が手戻り防止のためには重要であるが，追加で必要となる工数が課題となり，期間が限られている画面プロトタイピングでは実施が難しいという問題に対して，画面設計情報からのデータ構造自動抽出による自動生成ツールによって本番開発へ設計成果を移行することができないプロトタイプ開発作業を完全自動生成する手法を提案した．

また，本手法について自動生成ツールの試作と画面プロトタイピング開発プロジェクトへの試験適用によって実現性と有用性を評価した．

実現性については，画面プロトタイプに定義された設計情報により，プロトタイプ用アーキテクチャと本番アーキテクチャの各々に対する生成が確認できている．

有用性については，画面プロトタイプん開発期間について，想定するレビュー間隔内に開発を完了できることが確認できた．ロス工数については，全ての機能について開発者が手を加えることなく生成できたため，削減できたと考えるが，テストプロジェクトでは本番開発まで実施できなかったため，定量的な測定ができなかった．定量的な効果測定については本研究の今後の課題としたい．

6.2 今後の課題

今後の課題として，本手法の定量的な評価があげられる．ロス工数の削減率について，実プロジェクト適用による定量的評価の実施や，本手法を用いた群と用いなかった群の比較対象実験を実施したい．また，本研究では，画面の振る舞いなど画面自身に関わる開発の効率化は対象としていない．プロトタイピングのさらなる迅速化のためには画面振る舞いに関するプロトタイプ向け部品やモデル，パターンを用いた効率化が必要である．さらに，本手法を本番開発に応用した本番開発画面開発環境の構築や，プロトタイプで獲得した要求に基づくソフトウェアの効率的なテスト作成方法の開発などが課題として挙げられる．

謝辞 本論文の執筆にあたり，有益なご助言をいただいた日立製作所横浜研究所の吉村健太郎氏に深謝いたします．

参考文献

1) Nakakoji, K.: Interactivity, Continuity, Sketching, and Experience, *Proceeding of the 33rd international conference on Software Engineering*, Vol.2, p.621 (2011).
2) Dresselhaus, B.: Exciting new trends in design thinking, *Proceeding of the 33rd international conference on Software Engineering*, Vol.2, p.622 (2011).
3) Snyder, C.，黒須正明：ペーパープロトタイピング最適なユーザインタフェースを効率よくデザインする，オーム社 (2004).
4) 情報処理推進機構ソフトウェアエンジニアリングセンター：共通フレーム 2007―経営者、業務部門が参画するシステム開発および取引のために，オーム社 (2009).
5) 情報処理推進機構独立行政法人：ソフトウェア開発データ白書 2010-2011，独立行政法人情報処理推進機構 (2010).
6) Shore, J. and Warden, S.: アート・オブ・アジャイルデベロップメント，オライリージャパン (2009).
7) Adobe Systems Incorporated: Flex, `http://www.adobe.com/jp/products/flex/`.
8) Oacle Corporation: Java, `http://java.com/ja/`.
9) 株式会社日立製作所：HiRDB，`http://www.hitachi.co.jp/Prod/comp/soft1/hirdb/index.html`.
10) 小形真平，松浦佐江子：UML 要求分析モデルからの段階的な Web UI プロトタイプ自動生成手法，コンピュータソフトウェア， Vol.27, pp.14–32 (2010).
11) Bozzon, A., Comai, S., Fraternali, P. and Carughi, G.T.: Conceptual modeling and code generation for rich internet applications, *Proceedings of the 6th international conference on Web engineering - ICWE '06*, Vol.1, pp.353–360 (2006).