ソフトウェア工学 118－12
（1998．3．10）

社会情報システムのGUI設計指針の提案

藤巻 昇　上野 篤　岡安二郎　平山雅之
株式会社 東芝 研究開発センター S&S研究所

我々は、社会情報システムのGUI部の要求仕様を設計工程に迅速・正確に反映させる方法について研究している。ここで社会情報システムとは、不特定多数の利用者を対象に情報やサービスを与えるシステムを指す。

近年、社会情報システムが提供するサービスは急速に拡大中で、製品の市場投入サイクルは高速化している。それに伴い開発現場には顧客満足度を損なうこと無く迅速に製品を開発することが要求されている。しかし開発現場には、要求仕様から矛盾や抜けが取れない、要求仕様のデータが設計工程で十分再利用できていない、変更内容を設計へ反映するのに工数が掛かる、といった問題が存在している。

これらの問題点解決のため、我々はGUI部分の開発に焦点を絞り、プロトタイピング環境による効果的な要求獲得、要求獲得工程のデータの再利用、GUI部の独立性の確保、の3つの方法を提案している。

我々は、実システムの開発にこれらの方法を適用し問題解決に効果があることを確認した。また、適用対象を広げるにあたっての課題点を明らかにした。

A proposal of the graphical user interface design guidelines for social information systems

Noboru Fujimaki, Atsushi Ueno, Jiro Okayasu, Masayuki Hirayama
Systems and Software Research Laboratories,
Research and Development Center, TOSHIBA Corp.

We are studying a system development method for the graphical user interface of social information systems, which provide various services or information for the general public.

Services provided by the social information systems are increasing rapidly in recent years, and the market injection cycle of products is being shortened. Therefore, developers should develop products for short-term without spoiling customer satisfaction. There are, however, three problems in the GUI development as follows; the requirement tends to be inconsistent and incomplete, the requirement data isn't reused enough for design phase, changing GUI modules according to requirement costs too much.

In order to solve these problems, this paper proposes three GUI design guidelines: preparing an effective prototyping environment, reusing requirement data, and ensuring independency of GUI modules. We applied these guidelines to GUI development for the automatic teller machine and confirmed our guidelines effectiveness. In addition, we identified the necessary conditions of our approach for broader deployment.

1.はじめに

近年、銀行の現金自動取引装置(ATM[1])、飛行場の自動チェックイン機といった社会情報システムが普及してきている。これらのシステムが提供するサービスは現在急速に拡大中で、製品の市場投入サイクルは高速化し、顧客満足度の高い製品の迅速な開発が要求されている。社会情報システムはユーザインタフェース(以下UI)の出来がシステム評価と顧客満足度に直結する。またUIが要求仕様変更の集中する部分でもあるため、作り方が重要となる。しかし開発現場では要求仕様から矛盾や抜けが取れない、要求仕様のデータが設計工程で十分再利用できていない、変更内容を設計へ反映するのに工数が掛かる、といった問題が起きている。

以上の問題点解決のため、我々は特にGUI部分の開発に焦点を絞り、社会情報システムとそのGUI部の開発プロセスについて現状分析し、プロトタイピング環境による効果的な要求獲得、要求獲得工程のデータの再利用、GUI部の独立性の確保、の3つの方法を提案している。

また、これらの方法を実アプリケーション開発に適用した結果について報告する。

以下の議論を進める前に、用語の定義を行う。

画面遷移: 紙芝居的に画面が切り替わるその流れを指す。画面内の詳細な動作も含めてGUIの動作全体を画面の状態変化とし、これを画面遷移と呼ぶ場合もあるが、ここでは画面内の詳細な動作定義は「画面内動作」として区別する。

GUIの制御仕様: 画面のレイアウト情報を除く、画面遷移、画面内動作の情報をまとめてGUI制御仕様と呼ぶ。

GUIオブジェクト: 画面上に貼り付けられる「ボタン」「テキスト」といった部品で、表示のためのデータとアクションを持つ。コントロール、コンポーネントなどと呼ばれることもある。例えばOCX[2]はこれに当たる。

[1] Automatic teller machine

[2] Microsoft社が提唱するOLE2.0のオブジェクトとして作成されたコントロール

2.現状分析

本章では社会情報システムの特徴と、GUI開発プロセスを分析し、現状の問題点を明らかにする。

2.1.社会情報システムの特徴

社会情報システムの主な特徴としては以下の3点があげられる。

(1) GUIの出来がシステムの評価・顧客満足度に直結する

若年者から高齢者まで不特定多数の幅の広い利用者を対象にする。このためGUIには適切に利用者をゴールへ導くことが特に要求される。

(2)提供サービスが急速に拡大している

近年のテクノロジーの急速な進歩と市場オープン化・規制緩和に後押しされ、様々なサービスの自動化、新サービスの創出が進み、社会情報システムが提供するサービスの多様化が進んでいる。また、既存のシステムに仕様が追加されることが多い。

(3)GUIに画面遷移を持つ傾向がある

提供するサービスの多様化に伴い、操作も複雑化している。スムーズに操作させるには、手順を小さな工程に分割し、一手順ずつ示しながら処理を進める「画面遷移」の形態を採ることが多い。

2.2.GUI開発プロセス

社会情報システムのGUI部分の要求獲得工程と設計工程で行われる作業を理解する。ここでは以下に示す4つの主な工程があると捉える。各工程の目的、作業者、成果物は以下の通りである（表1参照）。

(i)要求獲得(1):要求の提示

目的は顧客からGUIに対する要求を吸い上げることである。作業は主に顧客とSEの二者間で行われる。得られる要求仕様には矛盾や抜けが含まれたり、断片的であることが多い。

(ii)要求獲得(2):要求の分析と厳密化

目的は、矛盾と抜けのない顧客の望む正確な要求仕様を獲得することである。作業は主にSEによって行われ、矛盾と抜けを校正する。また、要求仕様に関して開発者と実現可能性を議論し、必要なら顧客に細部の質問や動作の変更

を提案する。また、プロトタイピング等を使用し要求仕様の画面イメージや動作概要を顧客に正確に伝え、仕様に関して合意を得る。

(iii) 設計

目的は、要求仕様を満たすシステム構築のために、これを設計に変換し、詳細情報の追加を行うことである。作業は主に開発者によって行われる。画面遷移の設計書、画面内動作の設計書、画面レイアウト情報、システムと GUI 間のインタフェース仕様書が成果物となる。

(iv) 実装

目的は設計通りに正確に動作するシステムを実装することである。作業は開発者によって行われる。成果物は動作するシステムの GUI 部分である。

表1：要求獲得と開発の役割

		要求獲得(1)	要求獲得(2)	分析・設計	実装
作業対象者の役割の大きさ	顧客	大	中	−	−
	SE	大	大	小	
	開発者	−	小	大	大
成果物		矛盾・抜けを含む要求仕様	要求仕様書	画面遷移、画面内動作、インタフェースの設計書	動作するGUI部分のシステム
目的		顧客要求の吸い上げ	正確な顧客要求の獲得	要求仕様から設計への変換	顧客の望む動作するシステムの実装

2.3.GUI 開発で要求される点

2.1.節で示した特徴と2.2.節で示した開発プロセスを持つ社会情報システムの GUI 部分を開発するには、以下の要件が必要である。

(1) 開発サイクルを迅速に廻せること
(2) 要求獲得の段階で十分に使い勝手の評価が行えること
要求獲得用の支援環境が必要とされる
(3) 画面遷移を使用した開発が行えること
(4) 要求獲得の支援環境は SE にも使いこなせるものであること
既存システムへの仕様追加の場面が多い社会情報システムでは、要求獲得工程は SE 中心で作業が進む。
(5) 要求仕様の定義が厳密に行えること
使い勝手の評価だけでなく、設計工程に流す仕様を顧客の意図を正確に反映し、矛盾や抜けの無い正しいものにするための環境が必要である。

2.4.問題点

前節の要求は残念ながら十分に満たされていないのが現状である。これらの要件を阻害する問題点としては以下の３つが挙げられる。

(A)要求獲得(２)で矛盾や抜けが残る

要求獲得工程で作成される要求仕様が不正確であったり、詳細度が不足していることで、設計者が仕様を捉えきれず自前の知識でこれを補うため工数が掛かってしまう。また、完全な要求仕様がドキュメントに残らない、設計を行える人材の育成に時間が掛かるという副作用が生じる。この問題は主に GUI の制御仕様に関して起きる。

制御仕様が工程(ii)で十分に厳密に定義されない原因は次のものである。

画面遷移を持つ仕様であるに関わらず、要求仕様上では画面遷移と画面内動作が混在した状態で扱われる。この結果、次々に追加されるサービスで仕様が膨れると、整合性を取れなくなる。

制御仕様は、画面レイアウトとは別ドキュメントして作成されることが多く、画面レイアウトとの関連が取りにくい。

要求仕様が客先提出用の資料となってしまい、設計者の常識して記載されない要求仕様が存在する。

(B)要求獲得された情報が設計工程で十分再利用されていない

要求獲得工程のデータが設計工程で十分に再利用されない。 レイアウト情報はビジュアルプログラミングツールにより再利用されるようになった。しかし、GUI の制御仕様部分では未だ不十分である。

正確な GUI の要求獲得にはプロトタイピングツールは必要不可欠で、プロトタイプ後、制御仕様にあたるデータを再利用することが考えられる。市販のプロトタイピングツールの Rapid Plus™[3]のようにハードウェアとのコンカレント開発を考慮し、ソースコードの生成機能も持つものが存在するが、詳細な動作の定義を行う知識と高いスキルが要求される。また

[3] Rapid Plus は Emultek 社の登録商標です

Visual Basic®[4]（以下VB）では画面レイアウトの構築は簡易だが、プロトタイピングの動作をさせるには手作業によるコーディングが必要である。これらのツールでは、要求獲得作業をＳＥ中心で行うには敷居が高い。

その結果制御仕様はコメント的なドキュメントの形式を採ることが多く、データとしては再利用できない。

(C)仕様変更を設計に反映させるときに工数がかかる

GUIの仕様変更を設計・実装で反映するのに工数が掛かる主な原因の一つは、GUI オブジェクトとシステムの処理部分のモジュール結合度が高いことに起因している。近年ではコンポーネントウェア技術の進歩で汎用的なGUIコンポーネントが流通し、再利用が進み開発効率が向上している。しかし、コンポーネント同士、あるいはコンポーネントとシステムの制御部分とのインタフェースは、依然として人手によってコーディングされることが多い。そのため結合度は高くなり、変更範囲特定や変更作業自体を困難にしている。

3.提案するGUI構築指針

2.章で示した３つの問題点を解決するためにGUI構築環境の満たす要件と、GUI部の構築方法について指針を述べる。

まず、問題(A),(B)を解決する方法として、GUI構築環境が満たす要件を述べる。ここで、GUI構築環境とは要求獲得工程で利用されるプロトタイピングツール、設計工程で利用されるGUI構築ツールを含んでいる。

また(C)の問題を解決するためにGUI構築方法に関する提案を行う。以下詳細を述べる。

3.1.プロトタイピング環境

要求獲得を十分に行い、(A)の問題点を解決するために、まず目的とするプロトタイピングの範囲を明らかにし、プロトタイピング環境が備える要件を示す。

プロトタイピングの範囲は次の通りとする。

- GUIに対象を絞りGUI以外の部分とのインタフェースまでプロトタイピングする。
- GUIの要求具体化、操作性確認にフォーカスし、レスポンス等、性能関連の評価は範囲に含めない。

プロトタイピング環境の備える要件は次の通りとする。

- 本番システムにより近い形式でプロトタイピングを行えること。
- 画面遷移と画面内動作を明確に区別し扱えること。(ii)の作業の厳密化にはGUIの制御仕様の定義を整理して行う環境が必要で、性質の異なる仕様を明確に区別し定義させる事で解決する。
- 初心者でも使いやすい簡単な操作環境を備えること。

3.2.要求獲得工程のデータ再利用

開発サイクルを迅速に廻すには、データ再利用は重要なポイントとなる。(B)の問題を解決し再利用を行うため、プロトタイピング環境と設計環境が備える要件を以下に示す。

- レイアウト情報の再利用を行えること。
- プロトタイピングで使用した動作のデータを設計の制御仕様として再利用できること。
- 要求仕様データと設計データの整合が取れること。要求獲得工程と設計工程を繋ぐときに問題となるのは、要求仕様データと設計データの間で、お互いに行った変更を既存の相手のデータに反映させる方法である。

3.3.GUIの独立性の確保

設計工程の開発サイクルを迅速に行うにはGUIを独立性を高くするアーキテクチャを採用することがポイントである。

(C)の問題を解決しGUIの独立性を高くするポイントは以下の点である。

- インタフェースはメッセージベースで取ること。GUIとシステム間のインタフェースを、メッセージベースで取り、結合度を低くする。
- GUIの画面遷移と画面内動作を切り離すこと。メッセージベースのインタフェースは「画面遷移」「画面内動作」「その他のシステム」の３つの部分の間に存在させる（図１参照）。

画面内動作の実装は対象製品のプラットフォームへの依存度が高い部分である。一方、画面遷移はプラットフォームに依存しない。この二つを切り分けることで、画面遷移部分のポータビリティーを高く出来る。また、変更範囲が限

[4] Visual Basicは米国Microsoft社の登録商標です

定されるので、変更作業自体も省力化される。

●適用ドメインで必要なメッセージをパターン化し、記述の基準を設けることで記述レベルを合わせ、インタフェース定義時のガイドラインとする。パターンに則ってインタフェースの変更作業を行うことで、システム側との食い違いを減少できる。

なお、本アプローチを採る前提として、システム側もメッセージベースで処理が進むアーキテクチャを持つ必要がある。

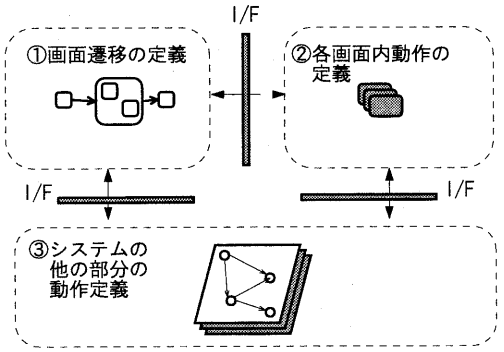

図1：各部のインタフェース

4.ATM への適用と評価

本章では、本提案を新型 ATM の開発に適用した事例について述べる。

4.1.適用

まず、適用の概要・目的を述べ、次に開発した GUI 構築環境の概要を示し、最後に３つの提案をどのように ATM ドメインに適用したか詳細を述べる。

4.1.1.適用の概要と目的

ATM の GUI 部の開発に本提案を適用する目的は、問題点の解決が行われることの確認、及び実適用した場合の課題の抽出である。

新型 ATM の制御ソフトウェア開発では、変更容易性と品質向上を目的にオブジェクト指向技術と CASE[5]技術を組み合わせたアプリケーション構築環境を開発した。ATM の GUI 構築環境は上記の一環として行ったものである。

なお、ATM の GUI の規模は画面数約 280 枚である。

4.1.2.GUI 開発の概要

GUI 開発環境のシステム構成と、この環境を使った作業の概要について述べる。

図 2に ATM の GUI 開発で想定したプロセスとプロダクトの関係を示す。また図 3は ATM の GUI 構築環境のシステム構成を表している。

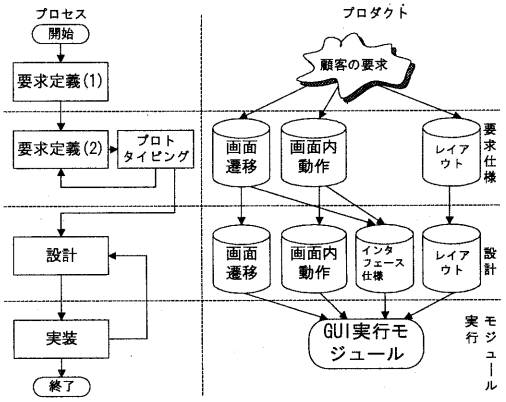

図 2：ATM の開発プロセスとプロダクト

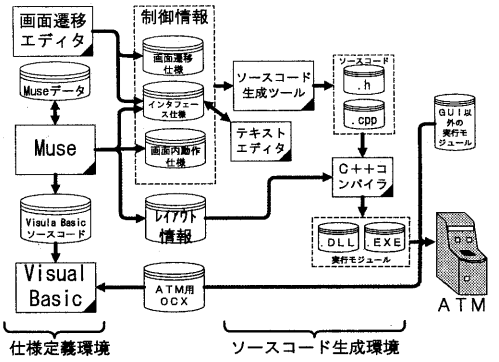

図3：ATM の GUI 構築環境

要求獲得工程と設計工程はどちらも共通のツールを利用している。図中では仕様定義環境の「画面遷移エディタ」「Muse」が該当し、それぞれ画面遷移仕様と、画面レイアウト及び画面内動作を編集する。SE が要求仕様を定義し、これに設計者が詳細な情報を肉付けし設計を行う（図 2のプロダクト部分参照）。

ソースコード生成環境は作成した制御情報とレイアウト情報を入力とし、ソースコードの生成、コンパイル・リンクを行う。

生成された GUI の実行モジュールと ATM ド

[5] Computer Aided Software Engineering

メイン用のライブラリ(OCX 等)、GUI 以外のシステムの実行モジュールをリンクさせて動作させることで ATM を制御する。

4.1.3.GUI 構築環境への適用

(A)ATM の GUI プロトタイピング環境

ATM の GUI 開発環境では Muse[6]を画面の要求定義用のプロトタイピング環境として利用している。図 4は Muse の編集イメージを示している。ここで Muse とは、マウス中心の操作で GUI の設計を行い、設計した画面の動作を即時プロトタイピングで確認できるラピッドプロトタイピングツールである。

画面内の動作の定義は GUI オブジェクト間にリンクを張り、リンク上に動作の為の情報を定義することで行う。

GUI 以外のシステムとのインタフェースの確認は、外部からのメッセージを発生させるボタンで模擬する機能を備えている。図 4の(a)で示した領域には外部からのメッセージを発生させるための仮想的なボタンを配置可能である。

Muse は、設計された GUI の仕様を VB コードに変換し、VB 上でプロトタイピングを行う。Muse では GUI オブジェクトの動作の記述を、対応する OCX の制御に変換することで VB コードを出力している [7]。この OCX は本番システムで使用するものと同一のものを使用した。

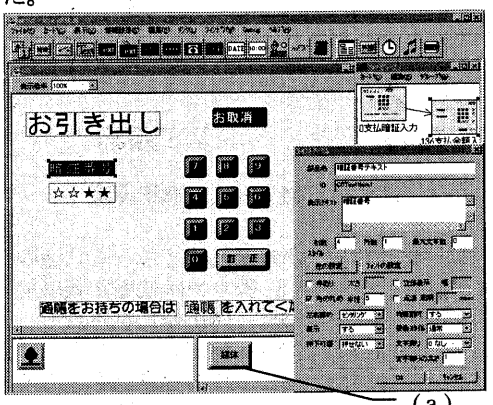

図 4：Muse 編集イメージ

[6] (Muse: Multimodal User-interface-design Support Editor;(株)東芝 マルチメディア研究所で開発されたラピッドプロトタイピングツール)

画面遷移の要求定義は画面遷移図エディタを用いて行う。図 5は画面をグループ化し設計情報として構成したイメージを示している。なお、遷移の為のイベントや条件はこの図では省略している。

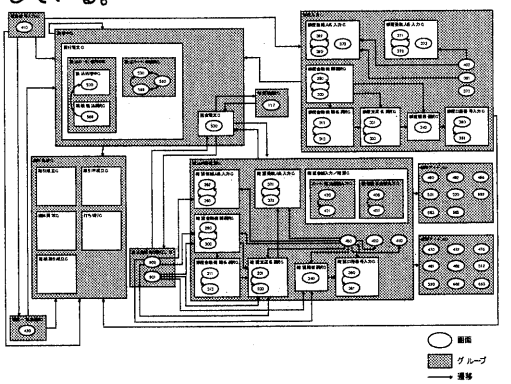

図 5：画面遷移情報のイメージ

(B)プロトタイピングのデータ再利用

データ再利用に関しては以下の点で実現した。

(1) 簡単なマウスによるリンクと選択の操作で画面内制御仕様を入力可能にした。
(2) 設計の仕様は要求仕様に肉付けする形で入力可能とした。

設計情報としては、
- ●動作を決めるシステムの条件
- ●GUI 外へのメッセージ送信
- ●外部のクラスライブラリの起動

といた詳細な情報である。
その結果ツールの形態としては、Muse に設計情報の入力を行える機能を追加し要求獲得と設計で同じツールを使用する構成とした。
(3) 入力された要求仕様データと設計データは同じファイルで扱った。この結果、変更反映の問題も解決できる。
(4) 実装用のコンパイラには Visual C++を使用する為、Muse から VC++用のレイアウト情報であるリソースファイルを出力した。

4.1.4.GUI 部の構築方法への適用

ここでは、GUI とシステムのインタフェースについて述べる。

ATM の制御アプリケーションはオブジェクト間のメッセージ交換をメカニズムとして持ったアーキテクチャを採用している。GUI 部

分もこのメカニズムに従ったメッセージ交換を行っている。

GUI とシステム間で必要なメッセージパタンを抽出し、インタフェース定義時のガイドラインとした。例えばシステムと画面遷移間のメッセージパターンは以下のようになった。

システム(A)と画面遷移(B)間のメッセージパタン
表示要求 (A)→(B)：画面を表示するための要求
要求キャンセル (A)→(B)：画面の表示要求のキャンセル
表示完了 (B)→(A)：表示要求により画面が実際に表示されたことを要求元へ伝えるためのメッセージ

これらのメッセージの受信－送信の対応はインタフェース仕様書により定義付けされているので、返信メッセージの宛先は自動的に指定される。つまり「表示要求」を受信したとき、これらのメッセージの送り主を記憶しておき、対応する「表示完了」メッセージの発信時に記憶した送り主に返送する。そのため、仕様記述時には送信先を意識する必要がない。この送信元の動的な管理機構を実装のメカニズムとして持つことで、GUI－システム間の結合度をより低く保つことが可能になっている。

4.2.評価

ATM 適用で３つの問題点を観点に評価を行う。

4.2.1. (A)要求獲得の厳密性

- 画面遷移と画面内動作を要求獲得段階から分けたことにより、要求仕様が整理され、漏れや誤りの確認が組織的に行えるようになった。また、要求仕様に変更が入った場合でも、変更範囲の特定が容易になった。
- 画面遷移図のモデルを取り入れたことで、従来は記述されていなかった画面遷移が視覚的に表現でき理解性が高まった。
- Muse のリンク機能で、画面の制御仕様は GUI オブジェクトと直接関連付けられるようになり、仕様の確認が容易になった。
- SE が短期間で容易に制御仕様の要求定義が行えるようになり、Muse を容易に導入可能なことが確認できた。

4.2.2. (B) データの再利用

- 従来コメント的に扱われていた制御仕様も設計工程で再利用可能となり作業効率が上がった。
- 要求仕様と設計のデータを同じファイルで管理することにより、データ間の整合は完全に取れるようになった。
- ソースコード生成ツールによって、作成された設計仕様からソースコードの自動生成が行えるので、要求獲得から実装までデータが再利用されるようになり、開発サイクルが迅速に廻せるようになった。例えば１画面に閉じた動作や GUI オブジェクトの物理的属性の変更なら、数分のオーダで GUI 実行モジュールの生成まで完了する。

4.2.3. (C) GUI の独立性

- ３つの I/F を持たせたことで、仕様変更による影響範囲が限定される。
- メッセージパターンのガイドラインで、規定されたメッセージ送受信を行っていることで、インタフェース仕様の定義作業が定式化される。
- お互いの仕様変更時に他に与える影響を、変更したメッセージの種類として容易に見積もれる。
- 実装のアーキテクチャ上、画面遷移と画面内動作部分を切り分けたことで、画面遷移部分のプラットフォームからの独立性が高くなった。
- 画面遷移と画面内動作を並行して開発可能となり効率的になった。

5.考察

前節で示したように、３つの提案は ATM ドメインへの適用に関しては効果的なことが確認できた。ここでは、社会情報システムの GUI 設計技術として汎用的な観点から考察を行い、さらに要求獲得と設計の接続をスムーズにし開発サイクルを迅速化するための要件について考察する。

5.1.要求獲得と設計で同じツールを使用することについて

ATM 適用では要求仕様と設計は同じツール上で入力が可能とし、SE･設計者間の入力情報に対する責任分担は運用でカバーしている。各適用ドメインのプロジェクト毎に責任分担が柔軟に行える点は利点である。しかし運用規則だけでは責任分担が不明確になったり、情報の漏れが生じる可能性が大きい。ツールにも作業

対象者ごとに入力モードを設定するなどの支援機能が必要となる。

5.2.画面遷移と画面内動作の境界について

画面遷移と画面内動作の仕様の境界は曖昧さがある。例えば、暗証入力を促す画面と、暗証再入力を促す画面ではテキスト数文字の違いしかなく、画面遷移、画面内動作のどちらに含めるか明確な指針は挙げにくい。仕様検討やバージョンアップの段階でこの境界の見直しが行われた場合、提案した方法では、画面遷移と画面内動作は切り分けられているので、それぞれ仕様を手作業で修正することになる。このような境界線の移動の検討も容易に行える仕組み、あるいは判断基準が必要である。

5.3.要求獲得(1)と要求獲得(2)のギャップについて

今回は要求獲得(2)以降にフォーカスしアプローチを行ったが、2.2.節の要求獲得(1)と要求獲得(2)の間にあるギャップを埋める作業の支援技術も今後考慮していく必要があるだろう。つまり、矛盾や抜けを含んだ断片的な要求仕様から、正確な要求仕様を得るための支援技術といったものである。

5.4.性能評価レベルのプロトタイピングについて

今回のプロトタイピングの範囲は操作性確認、要求具体化の手段としてプロトタイピングを用いたが、システムのレスポンスを含めるとより厳密な要求獲得が行える。手軽にレスポンスの評価が行えるような機能も必要とされる。

5.5.インタフェースの定義について

メッセージベースでインタフェースを定義する本方式は効果的であるが、これは仕組みを与えるものなので、インタフェースの定義を行う前にGUIとシステムの役割分担を明確に定義することが重要である。役割分担が不明確な場合、メッセージの擦り合わせに時間が掛かることがある。

6.おわりに

社会情報システムを対象にGUIの設計方法とGUI構築環境について3つの提案を行い、ATMの実機開発に適用した。その結果、ラピッドプロトタイピングツールMuseとCASE技術を組み合わせたGUI構築環境の実装が行え、要求獲得と設計の間に問題点解決に効果があることを確認できた。また要求獲得工程と設計工程の接続をよりスムーズにするための検討点を挙げることができた。

今後は、ATMとは異なる仕様の特徴を持った社会情報システム製品に適用を行い、仕様の規模、画面構成の特徴、製品の制約（メモリ、スピード）などによって、どの様な要件が必要となるかを検討する。例えば、より画面の構成が複雑で画面遷移と画面内動作の境界が付けにくい仕様を持った製品などが挙げられる。また、GUIだけでなくUI全体の設計方式に広げていきたい。

7.参考文献

[1] 「誰のためのデザイン？ -認知科学者のデザイン原論」新曜社、D．A．Norman著、野島久雄 訳、1990

[2] “A System of Patterns - pattern-oriented software architecture “,JOHN WILEY & SONS, F.Buschmann,R.Meunier,H.Rohnert,P. Sommerlad,M.Stal,1996

[3] オブジェクト指向最前線-情報処理学会OO'96シンポジウム，青山幹雄他，朝倉書店,1996. 7

[4] SEの基礎知識「アプリケーション開発技術」,布川薫他、リックテレコム、1997

[5] アプリケーション実行時GUIレイアウト変更機能、増田英孝、情報処理学会論文誌、Vol.35 No.9 1794-1806

[6] 社会情報システム向けSE支援システム、小尾ほか、東芝レビュー、Vol51 No.5 P59-P62,1996.

[7] ラピッドプロトタイピングツールMuseの開発、雨宮ほか、第54回情報処理全国大会、1-P.205-P.206,1997.3