筑波大学大学院博士課程

システム情報工学研究科修士論文

携帯端末上でコンテキスト依存プログラムを記述するためのビジュアルプログラミング環境

西本 裕貴

(コンピュータサイエンス専攻)

指導教員　田中 二郎

2009年3月

概要

携帯端末の機能やセンサを利用することによって，ユーザや携帯端末のコンテキストを取得することが可能になっている．取得したコンテキストを利用することで,「大学にいるときは，自動的にマナーモードにする」といったような動作を携帯端末に行わせることが可能になる．しかし，このような動作は，ユーザによって多種多様な要求があるため，ユーザが自分自身で定義できることが望まれる．

本研究では，ユーザの多種多様な要求に対応するため，ユーザが自分自身でコンテキストに依存して動作するプログラムの作成を可能にすることを目的とし，携帯端末上でコンテキスト依存プログラムを記述するための環境である MoCoPro を示す．MoCoPro は，ビジュアルプログラミング形式を採用し，プログラムを視覚的に表現することによって，携帯端末を使用するエンドユーザがプログラムを記述することを可能にする環境である．

プログラムの記述形式には，ECA ルールを採用し，プログラムをコンテキスト定義とアクション定義に分割して記述することで，エンドユーザであっても理解しやすい，簡易な記述方式を示す．コンテキスト定義には，処理を実行するためのコンテキストの条件を記述する．複数の条件を組み合わせることや，条件の間の関係を示すことで，単純な記述により複雑な条件を記述できる．アクション記述には，コンテキスト定義に示した条件を満たした際に実行する処理を記述する．携帯端末やネットワーク上の機能を組み合わせることで，ユーザの要求を満たす処理を記述することが可能になる．

また，携帯端末の入力インタフェースとして，タッチパネルを採用し，携帯端末の限られた画面の中で，ビジュアルプログラミングを行うための操作方法を示す．フォルディング操作は，複数のプログラムの要素を重ねて表示することにより，画面を広くすることや意味のある単位にまとめることを可能にする．クロッシング操作は，画面にボタンを配置することなく，素早い画面の切り替えを行うことを可能にする．

目 次

図 目 次

第1章　はじめに

本章では，本研究の背景，目的，貢献，論文の構成を示すとともに，題目に現れるコンテキストというキーワードについて説明する．

1.1　研究の背景

1.1.1　携帯端末の普及と発達

携帯電話は広く普及しており，ITU（International Telecommunication Union）[20] と総務省 [26] の調査によると，2008 年の統計では世界で 61.0%，日本では 84.7%の普及率となっている．携帯電話の高機能化も顕著であり，一般に普及するような機種であっても，通話やメールなどの基本機能に加えて，カメラ，インターネット接続，音楽再生，電子決済，PIM（Personal Information Manager）などの多くの機能を持つようになっている．また，携帯電話には，様々なセンサが搭載され始めている．例えば，2008 年 7 月に発売された iPhone 3G には，GPS（Global Positioning System）センサ，加速度センサ，環境光センサ，近接センサの 4 つのセンサが搭載されている．図 1.1 は，Parks Assosiates 社 [6] が行った GPS センサを搭載している携帯端末台数の推移予測である．2012 年まで 40 %近くの年間成長率で上昇し，2012 年には世界で 8 億 3400 万台を上回ると予測している．GPS センサから得られる位置情報は，ナビゲーションや子供の防犯対策などの様々な場面で利用することができる．このため今後も大きく成長することが見込まれているが，他のセンサについても，新たな利用方法が考案され，センサに対するユーザのニーズが高まれば，より多くの携帯端末に搭載されるようになると考えられる．

なお，携帯電話は高機能化により，PDA（Personal Digital Assistant）や PND（Personal Navigation Device）などの他の端末との境界が曖昧になってきている．このため，以降では携帯電話についても携帯端末と呼ぶ．

1.1.2　コンテキストとは

携帯端末の機能やセンサを利用することによって，ユーザや携帯端末の様々な状態を取得することが可能になる．例えば，携帯端末の時計やメールの機能を利用することで,「現在の時刻」や「誰からメールを受信したのか」といった情報を取得することができる．また，携

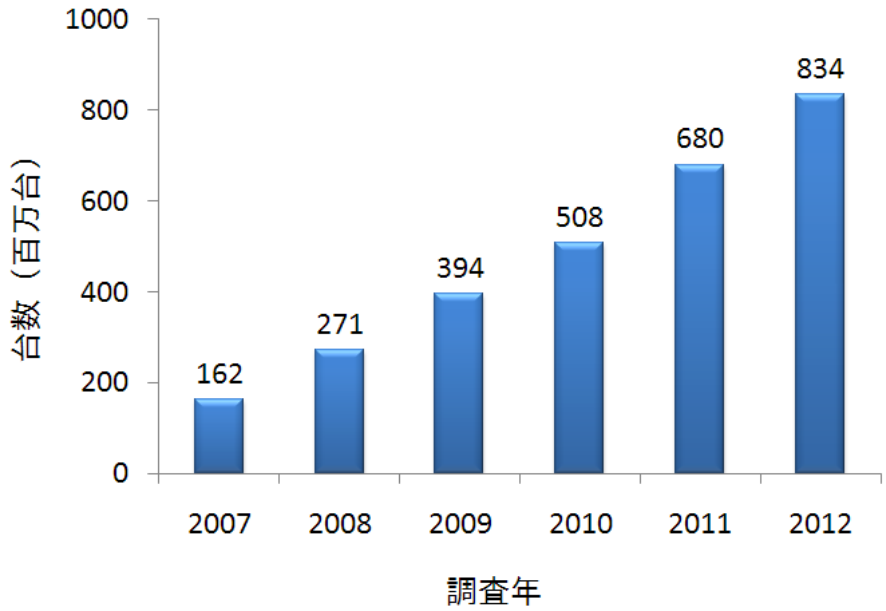

図 1.1: GPS センサを搭載した携帯端末台数の推移予測

帯端末に搭載されている GPS センサや加速度センサを利用することで,「茨城県つくば市天久保にいる」という位置情報や,「ユーザは走っている」という動作情報を取得することができる．このような情報をコンテキストと呼び，Anind ら は，コンテキストを以下のように定義している [2] .

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

コンテキストの種類は様々あり，Schmidt らがその分類を行っている [19]．Schmidt らによるとコンテキストは，以下のように人的要因と環境的要因の 2 つに大別され，それぞれ 3 つの小分類を持っている．

- 人的要因
  - ユーザ（趣味，感情，経歴など）
  - 社会的な立場（グループ，他者との関係など）
  - タスク
- 環境的要因
  - 場所（絶対位置，相対位置など）

– インフラ（通信環境，機器のパフォーマンスなど）
– 物理状況（時間，明るさ，温度など）

コンテキストを利用するシステムをコンテキストアウェアシステムと呼ぶ．Anind らは，コンテキストアウェアシステムを以下のように定義している．

> A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancey depends on the user's task.

コンテキストアウェアシステムの例を示すために，携帯端末上に買いたい本のリストを作成することを考える．携帯端末のメモ機能を用いて，単純にリストを作成した場合であると，本屋を訪れた際に，ユーザがリストの存在を思い出し，手動でリストを表示する必要がある．このため，本屋を訪れた際に本のリストを見ることを忘れることや，本屋の付近まで行ったにもかかわらず，本屋に立ち寄ることを忘れてしまうことがあると考えられる．しかし，買いたい本のリストが，場所のコンテキストを認識できるようになると，本屋を訪れた際や本屋付近を通りがかった際に，ブザーと共に自動的に買いたい本のリストを表示することで，リストを見忘れることを防ぐことができる．このように，コンテキストをシステムに認識させることにより，より使いやすく便利なシステムを実現することが可能になる．

現在，世界的に広まってきているユビキタスコンピューティングでは，ユーザが意識することなくコンピュータを利用できる環境が重要である．このため，生活の至るところにコンピュータやセンサを配置し，コンテキストアウェアシステムにより，ユーザの生活を便利にしようとする研究が多数行われている．コンテキストアウェアシステムの初期の研究として，Active Badge[22] や Cyberguide[1] がある．Want らの Active Badge は，ユーザに赤外線の信号を出力するバッジをつけ，各部屋に置かれた赤外線センサを用いてユーザの場所を認識することにより，ユーザがいる部屋に電話を転送することができる．Abowd らの Cyberguide は，携帯端末上の観光ガイドシステムであり，ユーザの現在位置やこれまでに訪れた場所の履歴などの場所のコンテキストを利用することによって，現在地周辺の情報やまだ訪れていない場所の推薦を行う．近年の研究では利用するコンテキストが増加する，コンテキストアウェアシステムが使用される場面が広がりを見せるなどの変化が起こっている．Yoon らは，携帯端末で撮影した写真に時間や場所，写っている人などの情報を付加し，その情報と同様のコンテキストになったときに，携帯端末の待ち受け画面やアプリケーションの背景にその写真を表示するシステムを開発し，携帯端末を使用する日常の何気ない場面でのコンテキストアウェアシステムを示している [25]．Agarwal らは，手術室において医者や患者，手術用具の状態を取得し，誤った手順で手術が行われることを防ぐことや，患者の状況や手術の進行に合わせて適切な情報を表示するシステムを開発し，安全性が問われる場面でのコンテキストアウェアシステムの可能性を示している [4]．このように我々の生活を便利にするコンテキストアウェアシステムは，今後も研究，開発が盛んに行われ，ユビキタスコンピューティングが実現する社会において当然のように利用されることになると考えられる．

1.1.3 携帯端末上のコンテキストアウェアシステムの問題点

コンテキストアウェアシステムの開発が多数行われるようになり，ユーザは既に開発されたコンテキストアウェアシステムを利用することで，ある程度は要求を満たすことが可能となってきている．しかし，ユーザの要求は多種多様であり，既存のシステムだけでは，ユーザの要求を満たせない場合がある．これは，利用できるコンテキストや機能，またその組み合わせに制限があるためであると考えられる．例えば，ユーザによっては，上述の買いたい本のリストに時間のコンテキストを認識させることにより，勤務時間中にリストを表示しないようすることや，買いたい本の商品紹介ページを表示することを望んでいるかも知れない．このようなユーザの要求は，低い次元では，電話が掛かってきた相手によって着信メロディを変更することや，季節によって待ち受け画面の画像の変更することなどに見られ，より高次に進むと，要求を満たすアプリケーションをダウンロードすることや，自身専用のアプリケーションを PC 上で開発することもある．

本研究では，ユーザの要求を満たすコンテキストアウェアシステムを，ユーザ自身がプログラムを記述することにより実現できるようにすることを目指す．本研究では，ユーザが記述するコンテキストを利用するプログラムをコンテキスト依存プログラムと呼ぶことにする．

1.2 研究の目的

本研究の目的は，コンテキストアウェアシステムに対するユーザの多種多様な要求に対応するため，ユーザが自分自身でプログラムを記述し，コンテキストアウェアシステムを開発することができる環境を提供することである．さらに，コンテキスト依存プログラムの記述を携帯端末上で行えるようにすることによって，ユーザがプログラムを作成したいと思ったときに，時間や場所を選ばずにプログラムを作成できるようにすることや，要求の変化に合わせてプログラムをすぐに変更できるようにすることを可能にする．

PC 上では，C 言語や Java などのテキスト形式のプログラミング言語を用いてプログラムを記述する．このような言語は，高い自由度を持っているため，様々な処理を行うプログラムを記述することができるが，携帯端末上で使用するには，画面が小さいことや入力インタフェースが貧弱であることが問題となるため適していない．また，携帯端末のユーザの大半がプログラムの知識を持っていないため，プログラムの知識が多く必要となるという点においても適していない．

そこで本研究では，ビジュアルプログラミング形式によりプログラムを記述することを考える．ビジュアルプログラミング環境は,「図形，アイコンなどの視覚的表現を用いてプログラムを表し，またそれらを操作することによってプログラムの編集や実行を行うことのできる環境」と定義されている [11][14]．ビジュアルプログラミングでは，図形やアイコンを操作することによりプログラムを記述するため，テキスト形式と比較してプログラムの構造をユーザが認識しやすくなる．加えて，コンテキストアウェアシステムを作成するために必要となるプログラムの知識を少なくすることや，効率よくプログラムの管理を行うことを可能にするための簡易なプログラムの記述形式を考える．また，ビジュアルプログラムの要素となる

図形，アイコンを操作するためのインタフェースを工夫することにより，携帯端末の画面の小ささや操作の難しさを軽減するインタフェースの構築を目指す．

1.3 研究の貢献

本研究の貢献を以下に示す．まず，記述形式に，ECA ルールを採用し，ECA ルールをビジュアルプログラミングにより視覚的に定義する方法を示した．また，ビジュアルプログラミングにより，複数のコンテキストの条件や，様々な機能を組み合わせた処理の記述の方法，またこれらを行うことに適した画面設計を示した．さらに，タッチパネル上でのクロッシング操作やフォルディング操作を示し，携帯端末上でビジュアルプログラムを記述することに適した操作を示した．

1.4 論文の構成

本論文の構成は次の通りである．本章では，本研究の背景と目的を示した．続く第 2 章では，携帯端末上でビジュアルプログラミングを記述する環境を構築する上での課題と，それを解決するための設計について述べる．第 3 章では，第 2 章で示した設計に従って実装した，携帯端末上でコンテキスト依存プログラムを記述するためのビジュアルプログラミング環境 “MoCoPro”[30][29] について述べ，第 4 章ではその実装について述べる．第 5 章では，MoCoPro の利用シーンについて述べる．そして，第 6 章で関連研究，第 7 章で研究の考察を行い，第 8 章でまとめる．

第2章　携帯端末上におけるビジュアルプログラミング環境の設計

本章では，まず本研究で開発するビジュアルプログラミング環境の設計方針について述べる．次に，設計方針に従って，設計を行ったプログラムの構成要素や記述形式，画面や操作インタフェースについて述べる．また，プログラムの記述方法や管理の設計についても述べる．

2.1　設計方針

MoCoPro の設計方針の 1 つ目は，携帯端末のみでプログラミングを可能にすることである．MoCoPro が目指しているのは，ユーザがコンテキストに関連して行うタスクに出会ったときに，直ちにその処理を行うプログラムを記述し，実行できるようにすることである．プログラムの記述を行うために，音声やジェスチャの入力や，PC や他の機器が必要になると，時間や場所を選ばず，直ちに使用することができるという携帯端末の最大の利点を失ってしまう可能性がある．本研究では，携帯端末の一般的なインタフェースのみを使用することで，プログラミングを可能にする．

MoCoPro の設計指針の 2 つ目は，携帯端末やネットワーク上の機能を組み合わせられるようにすることである．MoCoPro 上で記述するプログラムは，全く新しい機能を実現するものではなく，コンテキストに関連して日常的に行われているタスクを実現するようなプログラムである．本研究では，携帯端末やネットワーク上の既存の機能を組み合わせることや，処理を順番に実行できるようにすることで，ユーザの要求に合致するプログラムを記述できるようにする．

MoCoPro の設計指針の 3 つ目は，エンドユーザにも理解可能なように，簡易な記述形式や操作インタフェースにすることである．携帯端末を使用するエンドユーザの多くはプログラムの知識を有していないため，本研究では，C 言語や Java のように多量の知識を必要とするテキスト形式ではなく，ビジュアルプログラミング形式を用いることで，プログラムを視覚的に表現し，理解しやすくする．ビジュアルプログラミング形式を用いても，記述形式や操作インタフェースが複雑であると，エンドユーザに理解できないことに加え，携帯端末の小さい画面や貧弱な入力インタフェースなどの制約のため，プログラムの記述が困難となる．このため，携帯端末上であっても記述できる程度に簡易な記述形式や操作を行いやすいインタフェースが必要となる．

2.2 プログラムの構成要素

2.1 節で述べたように，プログラムの記述は，携帯端末やネットワーク上の機能を組み合わせることにより行う．本研究では，入出力インタフェースを持ち，入力されたデータを加工して出力するものを機能として考える．機能の実装は，専門家が行い，ユーザがプログラムを記述する上で使用できる 1 つのコンポーネントとして実現する．ユーザは，コンポーネントの入出力を接続していくことにより，データフロー図を描き，ユーザの要求に合致する機能を実現する．データフロー図は，データの流れを表した図であり，機能がどのように組み合わされているのかを，視覚的に示すことができるため，エンドユーザにも分かりやすいと考えられる．ユーザは，データフロー図を一定の記述形式に従い複数作成することで，プログラムを記述する．

2.3 プログラムの記述形式

プログラムの記述形式は，エンドユーザが理解することができる程度に簡易であることに加え，コンテキスト依存プログラムというドメインと親和性が高い形式であることが求められる．本研究では，ECA（Event-Condition-Action）ルールをコンテキスト定義とアクション定義の 2 つに分割して記述する形式を提案する．

2.3.1 ECA ルール

コンテキスト依存プログラムは，携帯端末の機能やセンサからの入力をイベントとして動作するプログラムである．同様にイベントにより動作するシステムとして，アクティブデータベースがある．アクティブデータベース [9] は，データベースに生じる挿入や削除，更新，検索などのイベントに対して，あらかじめ定められた処理を自動的に実行するデータベースである．ECA ルールは，アクティブデータベースにおいて，自動的に実行する処理を定義するためのルール定義である．ECA ルールは，以下の 3 つの要素からなる．

- **Event** ： ルールの実行のトリガとなるシグナル
- **Condition** ： ルールが実行された際に確認される条件
- **Action** ： 条件を満たした際に実行する処理

本研究では，この ECA ルールを用いて，プログラムを記述する．これは，ECA ルールの 3 つの要素の役割が簡易で明確であるため，エンドユーザにも理解可能だと考えられることに加え，アクティブデータベースというイベントを扱うシステムで使用されている実績があるためである．本研究では，ECA ルールをコンテキスト依存プログラムに合致するように，以下のようにマッピングする．

- **Event** ： プログラム実行のトリガとなる携帯端末の機能やセンサからの入力

- **Condition** ： プログラムが実行された際に確認される条件
- **Action** ： 条件を満たした際に実行される携帯端末やネットワーク上の機能

本来の ECA ルールと異なる点は，Event がデータベースに生じる挿入や削除などのイベントではなく，携帯端末の機能やセンサからの入力がトリガとなっている点と，Action が，携帯端末やネットワーク上の機能を利用する点である．1.1.2 節で述べた,「本屋の付近を通りがかったとき，買いたい本のリストを表示する」というプログラムを ECA ルールに当てはめると，Event が「GPS センサからのデータの受信」，Condition が「そのデータが本屋の付近である」，Action が「買いたい本のリストを表示する」となる．

2.3.2 コンテキスト定義とアクション定義

本研究では，2.3.1 節で述べた ECA ルールをコンテキスト定義とアクション定義の 2 つに分割して記述することによりコンテキスト依存プログラムを記述する．ECA ルールの Event と Condition に当たる部分はコンテキスト定義とし，処理を実行するために満たす必要があるコンテキストの条件を記述する．ECA ルールの Action に当たる部分をアクション定義とし，実行する処理を記述する．

コンテキスト定義

コンテキスト定義では，処理を実行するために満たす必要があるコンテキストの条件を記述する．このコンテキストの条件を記述したものをコンテキスト記述と呼ぶことにする．1.1.2 節で挙げたようなコンテキストを利用して，単体，もしくは複数のコンテキスト記述を組み合わせることや，コンテキスト記述間の関係を指定することにより，コンテキスト定義の記述を行う．コンテキスト記述の関係としては，複数のコンテキスト記述で示される条件をすべて満たさなければいけない「AND」と，複数のうちどれか 1 つのみを満たせば良い「OR」が考えられる．このような記述を行えるようにすることで，単純な条件の組み合わせで，複雑な条件を記述できるようにする．例を挙げると,「周りが暗い」という明るさのコンテキスト記述と「時間が深夜である」もしくは（OR）「時間が早朝である」という時間のコンテキスト記述,「携帯端末が置かれている」という動作のコンテキスト記述を組みあわせることで,「ユーザは寝ている」というコンテキストとすることが可能となる．また，機能を組み合わせることによる条件の記述を可能にする．例えば 1.1.2 節で述べた買いたい本のリストの例においても有用であり，買いたい本のリストに 1 つ以上の項目があるという条件を追加することで，買いたい本がないにも関わらず，処理が実行されるのを防ぐことが可能となる．

コンテキストは，携帯端末の機能やセンサから得られるデータをシステムが解釈することにより得られる．メールの受信や電話の着信，日時などの場合は，得られたデータがそのままコンテキストとして利用することができる．一方，加速度センサや環境光センサなどから得られるデータは，そのままでは単なる数値でしかなく，コンテキストとして使用できない．このため，そのデータがどのようなコンテキストを示しているのかを解釈する必要がある．し

かし，センサから得られるデータをコンテキストとして解釈するプログラムを記述する作業をユーザが行うことは難しい．コンテキストとして解釈するためには，数学的な知識が必要となることや，コンテキストごとの違いを調査する必要があるためである．このため，このような解釈は専門家があらかじめ行い，コンテキストを監視するコンポーネントとして実現することで，ユーザがコンテキストの条件を簡易に記述できるようにする．例えば，加速度センサの場合は，専門家がセンサから得られる 3 軸方向の加速度値の変化を動作のコンテキストとして解釈し,「走っている」,「歩いている」,「止まっている」といった状態を取得することができるコンポーネントにしてユーザに提供する．

コンテキストの監視を行うコンポーネントは，コンポーネントの種類によって監視するコンテキストの種類を示し，コンポーネントの入力によってコンテキストの条件を示す．コンポーネントは，入力として得られる条件と，現在のコンテキストを比較し，条件を満たしているかどうかを判別する．例えば，日時のコンテキストを監視するコンポーネントは，現在の日時が入力として受け取った 2 つの日時の間にあることを条件とする．

コンテキストを変化させるのは，携帯端末の機能やセンサからのイベントであるため，コンテキストを監視するコンポーネントを選択することで，そのコンテキスト依存プログラムが反応すべきイベントが定まる．すなわち，コンポーネントの選択が Event の選択に当たり，コンポーネントへの入力が Condition に当たる．機能を組み合わせて条件を記述した場合は，センサからの入力や電話の着信やメールの受信のように明確なイベントが発生しない場合があるため，一定時間間隔でイベントを発生させることにより，条件を満たしているかどうかを監視する．

アクション定義

アクション定義では，コンテキスト定義に記述された条件を満たした際に実行する処理を記述する．様々な機能を組み合わせることで，ユーザの要求に合致する何らかの機能を実現したものを，アクション記述と呼び，複数のアクション記述を順番に並べることで，処理を実行する順番を表す．

アクション定義で使用できるコンポーネントは，URL を入力するとその URL の Web ページを表示するブラウザや，タイトルと本文を入力すると新しいメモが作成されるメモアプリケーションなどが考えられる．また，このような機能への入力となるデータを加工するために，文字列の連結や四則演算などの機能についても提供する．

アクション記述の例としては,「アラームを起動する」や「スケジューラから取得した今日の予定と，メモから読み込んだ ToDo リストを連結し，今日行うべきこととして画面に表示する」のようなものが考えられる．そして，これらを順番に並べることで,「アラームでユーザに，今日行うべきことを表示したことを知らせる」という一連の処理を実現することができる．

2.4 画面設計と操作インタフェース

まず，画面設計について述べる．2.3.2 節において，プログラムの記述は，コンテキスト定義とアクション定義に分割して記述することを述べた．携帯端末の画面は小さいため，これらの定義は別々の画面を用いて記述を行うようにする．コンテキスト定義とアクション定義は，どちらもコンポーネントの入出力を接続し，データフロー図を描くことで，複数のコンテキスト記述やアクション記述を作成し，定義の記述を行う．コンテキスト記述やアクション記述は，他の記述との組み合わせや順番の関係があるため，同一の画面内で行えることが望ましい．このため，画面を複数に区切り，1 区切りごとに 1 つの記述を作成することにする．記述が画面より大きくなった場合は，スクロールすることにより，表示部分を切り替えられるようにする．

次に，操作インタフェースについて述べる．本研究では，2.1 節でビジュアルプログラミング形式でプログラムを記述することを既に述べたが，キーボタンを中心とした入力インタフェースを持つ携帯端末では操作に非常に手間がかかると考えられる．ビジュアルプログラミングでは，2 次元以上の環境に配置された要素を選択する操作を行うが，キーボタンは，上下左右のように特定の方向へ，1 回の操作で単位ごとに移動することが多いため，何度もキーを操作して要素を選択しなければならない．また，キーに操作を割り当てることを行っても，ユーザにキーと操作の対応付けの記憶を強いることになるなどの問題がある．

そこで，本研究では，タッチパネルを搭載した携帯端末を利用する．タッチパネルを搭載した携帯端末は，これまでのキーボタンを中心とした携帯端末と比較して，図やアイコンを直接操作できるという利点がある．また，マウスで行うようなクリックやダブルクリック，ドラッグ，ホイール操作の操作に対応し，タップ（画面を突つく），ダブルタップ（画面を 2 回続けて突つく），ドラッグ（タップした後にスワイプ），スワイプ（画面をなぞる）を行うことも可能である．これらの操作を用いることで，マウスで操作する際と同程度の操作性をもってビジュアルプログラミングを行うことが可能となる．しかし，タッチパネル固有の問題もあり，例えば，指により目標が隠れてしまい操作がしづらくなるという問題がある．この問題に関しては，目標を指のある位置からずらして表示することや，目標を操作できる面積を大きくすることで取り組む．また，マウスの場合と同様に，小さな目標は選択しづらいという問題がある．これに対しては，Parhi らのタッチパネル上における目標の大きさに関する実験 [15] を参考に，目標が一定以上の大きさになるようにする．さらに，携帯端末には，画面が小さいという問題があるため，それを考慮した操作インタフェースがより重要となる．

まず画面の切り替え時に行う操作について考える．画面全体，部分に関わらず，画面の切り替えを行う場面は多い．画面を切り替えるためのボタンやスクロールバーを画面上に配置することが考えられるが，そのために画面が小さくなってしまうという問題がある．そこで本研究では，クロッシング手法 [3] を導入する．クロッシング手法は，Accot らによって提唱されたコマンド実行のための手法であり，ポインティングデバイスがボタンやメニューなどの標的を横切ることによりコマンドを実行する操作体系である．本研究では，タッチパネルの端を指で横切ることにより，画面の切り替えを行い，画面を小さくすることなく，速やかな切り替えが行えるようにする．

次に，データフロー図が大きくなったときの場合を考える．テキスト形式のプログラムでは，プログラムをクラスや関数などに分割することで，1つ当たりのプログラムを小さくし，プログラムの流れを把握しやすくすることや，プログラムを意味のある単位に分割する．データフロー図を記述する際にも，同様のことが行えるが，小さなデータフロー図を別の画面で定義することになると，画面の切り替えが多くなり，プログラムの流れを把握しづらくなる．本研究では，データフロー図の一部を重ねて表示するフォルディング操作を導入する．フォルディング操作により，画面を切り替えることなく，データフロー図が画面を占領する面積を小さくし，フォルディングしたコンポーネントを利用することで，関数の作成を行えるようにし，意味のある単位としてコンポーネントをまとめることを可能にする．

2.5 プログラムの記述方法

コンポーネントは種類によって，文字列や数値，位置や時間などの様々なデータを出力する．このため，どのようなコンポーネント同士であっても接続できると，誤った記述を作成できてしまうことになる．これを防ぐために，コンポーネントの入出力に型を設けることで，同一の型を持つコンポーネント同士のみを接続できるようにする．また，コンポーネントの入力の数や型をコンポーネントの種類ごとに固定する．そして，その数に応じたプレースホルダーをコンポーネントの入力に接続した状態から始め，このプレースホルダーを埋めていくことにより，プログラミングを進めていく方式をとる．これにより，コンポーネントの接続にかかる手間をなくすことや，プログラムを記述する必要がある場所を分かりやすくすることが可能となる．さらに，コンポーネントを自動配置することで，コンポーネントの位置を調節する手間をなくす．

2.6 プログラムの管理

時間に伴ってコンテキスト依存プログラムに要求する処理が変化することや，思い通りの処理が実行されなかった場合など，コンテキスト依存プログラムを編集したい場面があると考えられる．ユーザの要求に応じてプログラムを変更できるようにするため，コンテキスト記述やアクション記述単位での編集や追加や削除を行えるようにする．

第3章　携帯端末上におけるビジュアルプログラミング環境：MoCoPro

本章では，MoCoPro が提供する機能や操作方法について説明する．まず，プログラムを記述する画面である定義画面について説明し，次に，プログラムの構成要素であるコンポーネントとコンポーネントを選択する際に使用するコンポーネントセレクタの説明を行う．続いて，MoCoPro が提供する機能である，プログラム作成機能，プログラム管理機能，プログラム実行機能をそれぞれの画面とともに説明する．

3.1　システムの概要

携帯端末上でコンテキスト依存プログラムを記述するためのビジュアルプログラミング環境である MoCoPro（Mobile Context Programming）の開発を行った [30][29]．MoCoPro は，コンテキスト依存プログラムの作成，管理，実行を携帯端末上で行える環境である．図 3.1 に MoCoPro を使用している際の様子を示す．

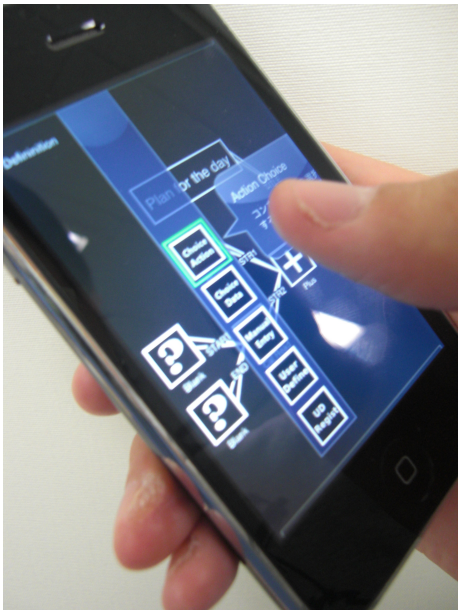

図 3.1: 携帯端末上で動作する MoCoPro

3.2 定義画面

図 3.2 に定義画面の構成を示す．図 3.2 上がコンテキスト定義画面であり，図 3.2 下がアクション定義画面である．どちらの定義画面においても，左から右に向かって複数のセルが並んでおり，1 つのセルに 1 つのコンテキスト記述やアクション記述を作成する．初期状態では，各セルの中央にブランクコンポーネントが配置されており，ユーザは，このブランクコンポーネントを始点としてセルに記述を作成していく．セルの数は，初期状態では 3 つであるが，すべてのセルを定義で埋めると新たなセルが末尾に追加される．

コンテキスト定義画面のセルの区切り線上には，隣接する 2 つの記述の関係を示す関係ボタンが表示される．この関係ボタン上をタップすると，関係ボタンの表示が AND と OR に切り替わる．関係ボタンを OR にすると，セルの区切りが暗く表示されるようになり，2 つのセルの記述のどちらかを満たせばよい OR の関係になる．関係ボタンを AND にすると，他のセルの記述をすべて満たさなければならない AND 関係になる．

それぞれの画面には，図 3.3 のように画面の上下にクロッシング操作が可能なことを示す影が表示されており，影が表示されている画面の辺を外側から内側に向けてクロッシング操作することによりコンテキス定義画面とアクション定義画面を切り替える．

3.3 コンポーネントとコンポーネントセレクタ

プログラムの構成要素であるコンポーネントと，コンポーネントを選択する際に利用するコンポーネントセレクタについて説明する．また，コンポーネントの一種であるコントロールコンポーネントにより実現される制御構文についての説明を行う．

3.3.1 コンポーネント

コンポーネントは，何らかの処理を行う機能を，プログラムで利用可能な要素として実現したものである．0 個以上の入力と，0 または 1 つの出力を持ち，有向なコネクタによって互いに接続される．コンポーネントの左側に接続されるコンポーネントが入力となり，右側に接続されるコンポーネントが出力となる．コンポーネントの種類ごとに入出力の数や型は固定されている．コンポーネントの下には，コンポーネントの名前が表示され，コネクタ上にはそのコネクタを流れるデータの説明が表示される．

コンポーネントには，オペレーションコンポーネント，コンテキストコンポーネント，データコンポーネント，コントロールコンポーネント，ファンクションコンポーネントの 5 種類がある（図 3.4）．

オペレーションコンポーネント

矩形領域内のアイコンで象徴される処理を行うコンポーネントである．例を挙げると，入力されたタイトルとメッセージを画面に表示する Dialog コンポーネントや，入力された 2 つの文字列を連結した文字列を出力する Plus コンポーネントなどがある．

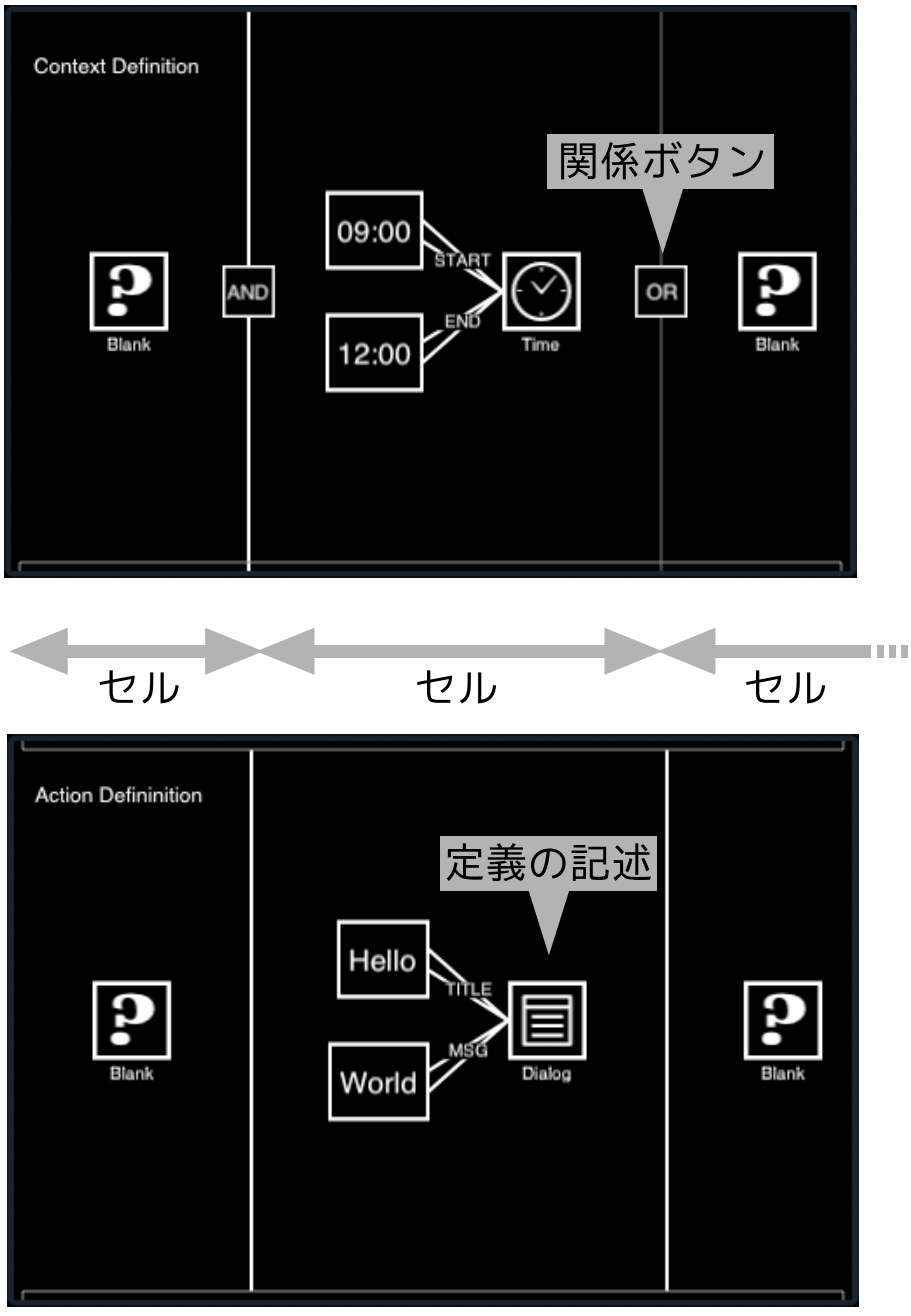
Context Definition
関係ボタン
09:00
START
END
12:00
Time
AND
OR
Blank
Blank
セル
セル
セル
Action Defininition
定義の記述
Hello
TITLE
MSG
World
Dialog
Blank
Blank

図 3.2: 定義画面の構成

図 3.3: クロッシング操作が可能なことを示す影

コンテキストコンポーネント

矩形領域のアイコンで象徴されるコンテキストの監視を行うコンポーネントであり，入力としてコンテキストの条件を受け取る．コンテキストコンポーネントは，コンテキスト記述においてのみ使用可能である．例を挙げると，図 3.4(b) に示すような Time コンポーネントがあり，現在の日時が入力して受け取った 2 つの時間の間であるかどうかを監視する．

データコンポーネント

矩形領域内に記述されているテキストが示すデータを出力するコンポーネントである．例としては，特定の URL や，現在の時刻などがある．

コントロールコンポーネント

条件分岐やループを実現するコンポーネントである．内部にプログラム片を保持し，条件によって実行するプログラムを変更することや繰り返し実行を行う．

ファンクションコンポーネント

ユーザが，フォルディング操作により複数のコンポーネントをまとめて，新たな 1 つのコンポーネントとして定義したコンポーネントである．ファンクションコンポーネントは，定義した後はオペレーションコンポーネントと同様の働きをする．

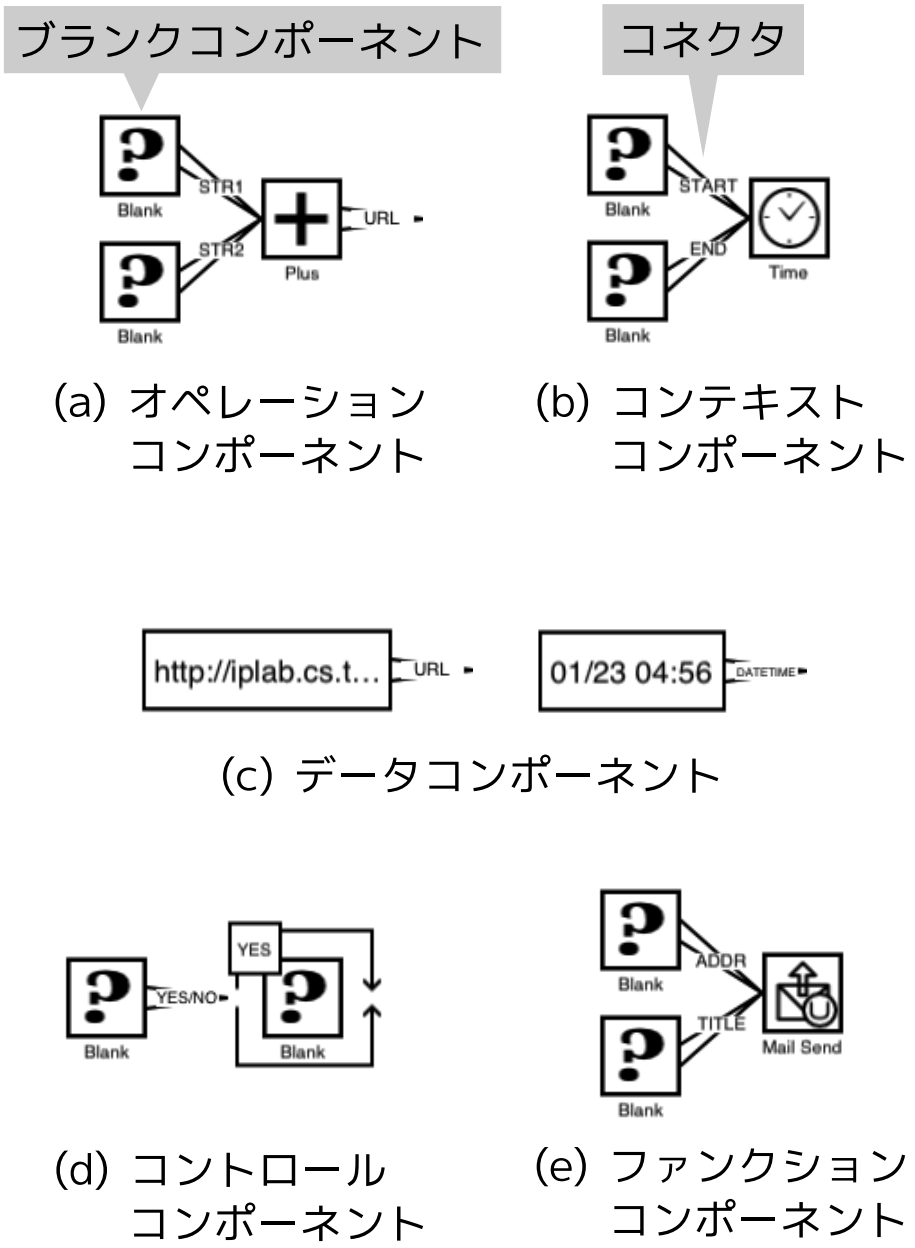

図 3.4: コンポーネントの種類

3.3.2 コンポーネントセレクタ

コンポーネントセレクタは，コンポーネントの階層リストであり，定義画面にコンポーネントを配置する際に使用する．コンポーネントセレクタには，コンポーネントが縦方向に並べられており，リストを上下にスクロールし，緑の選択枠内にコンポーネントを入れることで，コンポーネントが選択可能になる．選択枠内にコンポーネントが入れられると，横にある吹き出しの中にそのコンポーネントの説明が表示される．また，リスト内のコンポーネントに入出力がある場合は，その数に応じて，コンポーネントの左右に白いでっぱりが表示される．コンポーネントセレクタは，定義画面に配置されているコンポーネントをタップすることで表示される．コンポーネントセレクタには，コンポーネントを配置しようとしている位置の出力型に基づいて，その型を出力するコンポーネントのみが表示される．コンポーネントセレクタからコンポーネントを選択すると，タップしたコンポーネントがコンポーネントセレクタで選択したコンポーネントに置き換わる．コンポーネントセレクタからコンポーネントを選択するかコンポーネントセレクタが表示されている部分以外の画面上をタップすることで表示が消える．

現在の実装におけるコンポーネントの階層の変化を図 3.5 に示す．(a) のコンポーネントセレクタの 1 階層目は，コンポーネントの入力方法を選択する階層であり，上から「コンポーネントを選択する」,「データを手入力する」,「電話帳からデータを選択する」,「ユーザ定義のコンポーネントから選択する」となる.「コンポーネントを選択する」を選択した場合は，(b) のようにコンポーネントのリストが表示される．リストからコンポーネントを選択し，選択したコンポーネントに，複数の形態がある場合は，(c) に示すように 3 階層目に進む．例の場合であると，2 階層目の Calendar Read コンポーネントには，入力した 2 つの日時間の予定を取得するものと，入力した日付の予定を取得するものがあるため，3 階層目に移動することになる．

「コンポーネントを選択する」を選択した場合は，オペレーションコンポーネントやコンテキストコンポーネント，コントロールコンポーネントをリストから選択することができる.「データを手入力する」は，データコンポーネントをユーザの手入力により作成する場合に使用する.「電話帳からデータを選択する」は，電話帳からデータを引用したい場合に使用する.「ユーザ定義のコンポーネントから選択する」は，ユーザが作成したコンポーネントを選択する際に使用する．

データの手入力

コンポーネントセレクタの 1 階層目から「データを手入力する」を選択すると，ソフトキーボードから文字列を入力することや，リストからデータを選択することによって，ユーザがデータを手入力し，データコンポーネントを生成することができる．コンポーネントを配置しようとしている位置の出力型に基づいて，その型を入力することに適したエディタが表示される．

例として図 3.6 に，文字列，日時，場所を手入力する際のエディタを示す．文字列のエディ

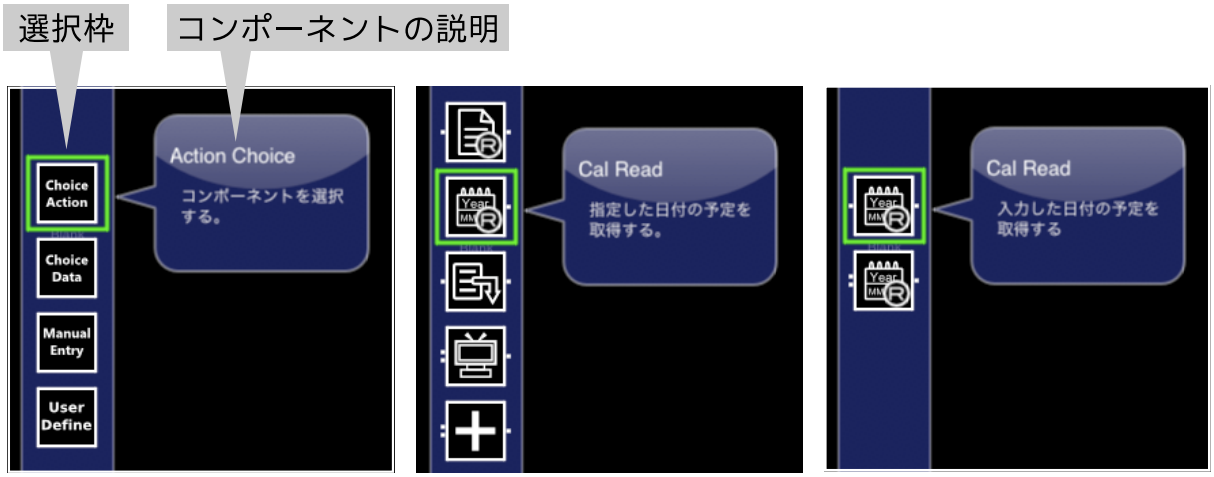

図 3.5: コンポーネントセレクタの階層ごとの変化

タは，ソフトキーボードから文字列を入力することにより文字列を出力するデータコンポーネントを作成する．日時の場合は，画面下部のリストから作成したい日時を選択することで，日時を出力するデータコンポーネントを作成する．場所の場合は，地図上の領域を選択することにより，場所を出力するデータコンポーネントを作成する．

また，日時や場所のエディタでは，アクション定義を実行する際の日時や場所のデータを出力するデータコンポーネントを生成することもできる．

電話帳からのデータの入力

コンポーネントセレクタの 1 階層目から「電話帳からデータを選択する」を選択すると，電話帳からデータを選択し，データコンポーネントを生成することができる．電話帳には，名前，電話番号，メールアドレス，住所，誕生日，URL，メモなどのデータを登録することが可能である．電話帳から選択する際も，コンポーネントを配置しようとしている位置の出力型に基づいて，配置可能なデータのみが表示される．図 3.7 に，電話帳からデータを選択している際の画面の様子を示す．電話帳に登録されている人を選択すると，その人のデータが表示されるので，その中からデータコンポーネントとして配置したいデータを選択する．例えば，メールアドレスを選択すると，選択したメールアドレスを出力するデータコンポーネントが配置される．

図 3.6: 文字列，日時，場所の手入力エディタ

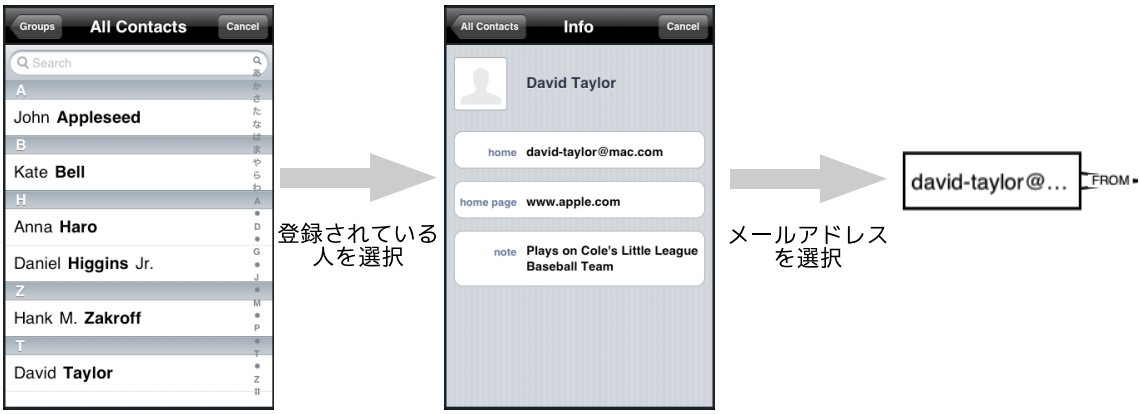

図 3.7: 電話帳からのデータの選択

3.3.3 制御構文

制御構文は，コントロールコンポーネントにより実現される．図3.8に条件分岐を実現する条件分岐コントロールコンポーネントの外観を示し，図3.9にループを実現するループコントロールコンポーネントの外観を示す．コントロールコンポーネントは，内部にプログラム片を保持し，条件によって実行するプログラム片を変更することや，繰り返し実行を行う．コントロールコンポーネントでは，内部のプログラム片の最も右側にあるコンポーネント（図3.8では，Plusコンポーネント，図3.9では，Mail送信コンポーネントがこれに当たる）の出力がコントロールコンポーネントの出力となる．

条件分岐コントロールコンポーネントは，入力として得られるブール値によって実行するプログラム片を変更する．左上にYESもしくはNOと表示された矩形領域上をタップすることで，それぞれのブール値時に実行するプログラム片の表示を切り替えることができる．図3.8左では，条件分岐コンポーネントの入力がYESの場合の処理を示しており，図3.8右では，条件分岐コンポーネントへの入力がNOの場合の処理を示している．

ループコントロールコンポーネントは，入力として得られるリストの要素を順番に取得して処理することや，入力して得られた数値の回数繰り返し実行することを行う．ループコントロールコンポーネントの内部では，コンポーネントリストにリストの要素や現在の繰り返しの回数を出力するコンポーネントが出現する．図3.9左では，ループコントロールコンポーネントの入力として，URLのリストを受け取り，リストの要素をブラウザへの入力とし，複数のWebページを開く処理を示している．図3.9右では，ループコントロールコンポーネントの入力として，繰り返し回数を入力し，繰り返しブザーを起動することで緊急性の高い用件に使用できる処理を示している．

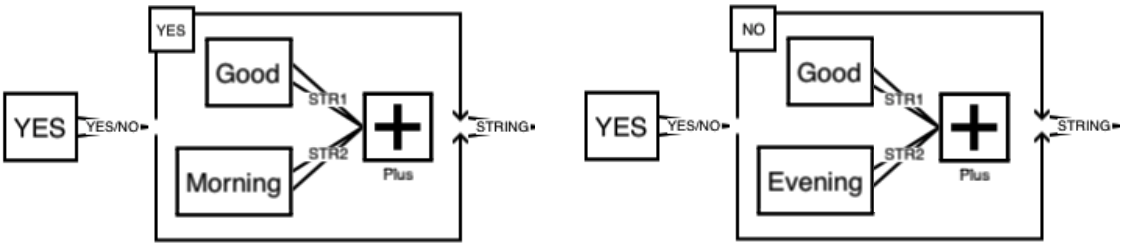

図 3.8: 条件分岐コントロールコンポーネントの外観

3.4 プログラム記述方法

プログラムは，コンポーネントの入出力を接続し，データフロー図を描くことにより記述を行う．プログラムの記述は，図3.10(1)のように，画面に配置されているブランクコンポーネントを起点として始める．図3.10(1)のブランクコンポーネント上をタップすると，図3.10(2)

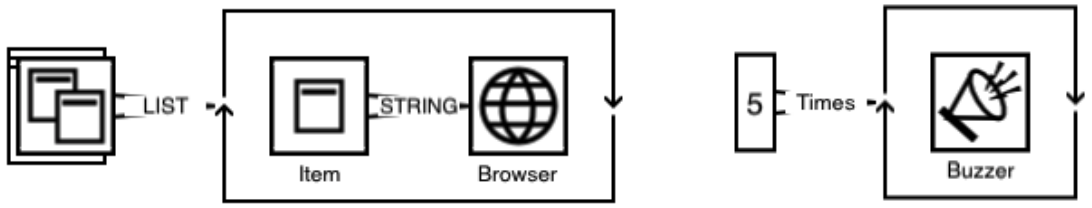

図 3.9: ループコントロールコンポーネントの外観

のように，コンポーネントセレクタが表示される．最初に配置できるコンポーネントは，出力のないコンポーネントのみであり，コンポーネントセレクタにおいても出力のないコンポーネントのみが表示される．図 3.10(2) から，1 つのコンポーネント（図 3.10(2) の場合は Dialog コンポーネントを選択している）を選択すると，図 3.10(3) のように，コンポーネントが画面に配置される．この時，選択したコンポーネントの入力に当たるコンポーネントがブランクコンポーネントとして，同時に配置される．続く記述は，図 3.10(4)，図 3.10(5) のように，新たに出現したブランクコンポーネントを埋めることによって進めていく．最終的に，データフロー図における，すべてのブランクコンポーネントを埋めると記述が完成したことになる．各記述において，最も始めに配置される，出力のないコンポーネントをルートコンポーネントと呼ぶ．

3.5 画面構成

MoCoPro は，大きく分けて 4 つの画面より構成される．4 つの画面とは，メインメニュー画面，プログラム作成画面，プログラム管理画面，待ち受け画面である．図 3.11 に示すメインメニュー画面は，他の画面の起点となる画面である．プログラム作成画面，プログラム管理画面，待ち受け画面は，それぞれ，プログラム作成機能，プログラム管理機能，プログラム実行機能に対応した画面である．以下では，それぞれの機能を画面とともに説明する．

3.6 プログラムの作成機能

プログラムの作成は，コンテキスト定義とアクション定義を記述することにより行う．定義の記述は，コンポーネントを接続し，データフロー図を描くことにより行う．

3.6.1 画面構成

プログラム作成画面は 3 つの画面より構成される．3 つの画面とは，コンテキスト定義画面，アクション定義画面，メニュー画面である．

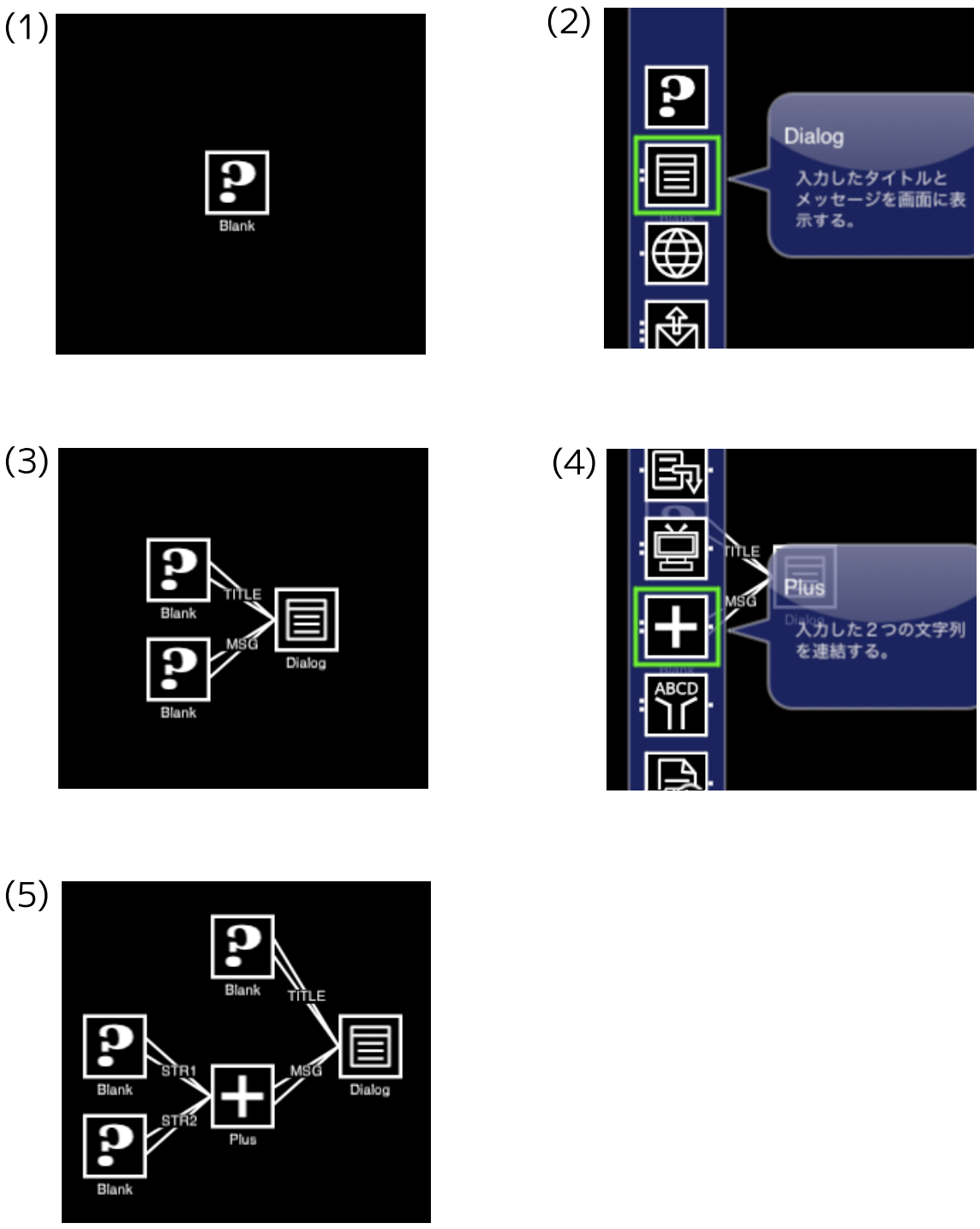

図 3.10: プログラム記述の流れ

図 3.11: MoCoPro のメインメニュー画面

コンテキスト定義画面は，コンテキスト定義を記述する画面であり，アクション定義画面は，アクション定義を記述する画面である．メニュー画面には，記述したプログラムの保存するためのボタンや，メインメニュー画面に戻るためのボタンが配置されている．

また，プログラム作成画面では，携帯端末の向いている方向により，縦画面と横画面を切り替えることが可能である．

3.6.2 コンポーネントに対する操作

コンポーネントに対する操作としては，フォルディング操作，コンポーネントのピン留め，コンポーネントの関数化，コンポーネントの移動，コンポーネントの機能の実行がある．コンポーネントをダブルタップすることで，フォルディング操作が行われ，長押しすることでコンポーネントメニューが表示される．図 3.12 に長押しした際に表示されるコンポーネントメニューを示す．コンポーネントメニューの項目をタップすることにより，コンポーネントのピン留め，コンポーネントの関数化，コンポーネントの移動，コンポーネントの機能の実行の各操作を実行することができる．

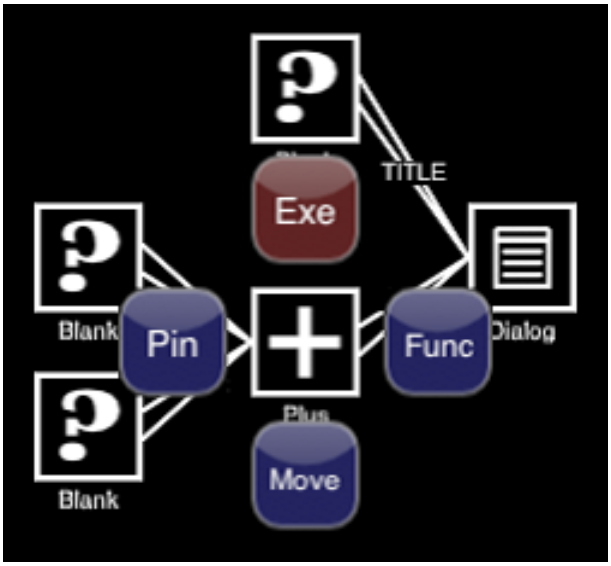

図 3.12: コンポーネントメニュー

フォルディングによる関数化

コンポーネント上をダブルタップすると，そのコンポーネントが起点となり，入力として接続されているコンポーネントが起点コンポーネントの下に重ねられる．この操作をフォルディング操作と呼ぶ．この際，ブランクコンポーネントとフォルディングされないようにユーザが指定したコンポーネントはフォルディングされない．起点コンポーネントをもう一度ダ

ブルタップすると，フォルディングされたコンポーネントは展開され，元の状態へ戻る．フォルディングされないように指定するにするためには，コンポーネントメニューから，ピン留め（Pin）を選択する．

例として，図 3.13 に，フォルディング前と，フォルディング後のデータフロー図の外観を示す．(a) は，あるメールアドレスに「Auto Replay!」というサブジェクトで，Busy というタイトルのメモを本文としたメールを送信するアクション記述の例を示している．(a) の MailSend コンポーネント上をダブルタップすると，(b) のようにフォルディングされる．この際，MailSend コンポーネントの入力のメールアドレスに当たるコンポーネントはブランクコンポーネントになっており，また，Memo 取得コンポーネントの入力のタイトルに当たるコンポーネントは，ピン留めされているため，フォルディングされていない．

また，フォルディングの起点コンポーネントを利用して，関数を作成することができる．関数の入力は，起点コンポーネントをフォルディングした際に，フォルディングされなかった，ブランクコンポーネントやピン留めしたコンポーネントとなる．関数の出力は，フォルディングの起点コンポーネントの出力が関数の出力となる．

関数を作成するために，コンポーネントメニューから関数の作成（Func）を選択すると，関数の名称と説明を入力するエディタが表示される．名称と説明を入力すると関数が登録され，起点コンポーネントが新しく作成されたコンポーネントに置き換えられる．関数のアイコンは，起点コンポーネントアイコンにユーザが定義した関数であることを示す「U」のマークをつけたものとなる．(c) は，(b) に示す定義を関数に置き換えた様子を示している．関数の入力は，(b) でフォルディングされなかったメールアドレスとメモのタイトルとなっている．

登録されたコンポーネントは，コンポーネントセレクタ 1 階層目の「ユーザ定義のコンポーネントを選択する」から選択できるようになる．

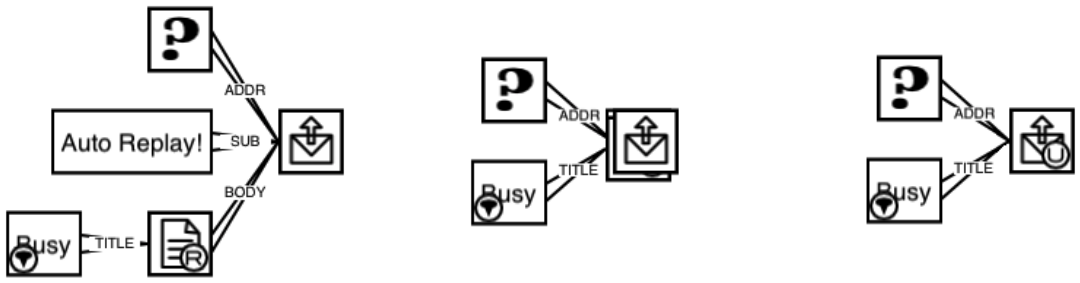

図 3.13: フォルディングと関数化

ドラッグアンドドロップによるコンポーネントの移動

コンポーネントメニューから Move を選択し，ドラッグすると，入力コンポーネントとともに，コンポーネントが移動し，他のコンポーネント上にドロップすることで，コンポーネ

ントの移動を行うことができる．ドロップすることが可能なコンポーネントは，ドラッグしているコンポーネントの出力型によって制限されている．ドラッグをしている際は，ドロップ不可能な先のコンポーネントは，薄く表示され，ドロップできないことを示す．ドロップ可能なコンポーネント上以外でドラッグをやめた場合には，元の場所に戻される．指によりコンポーネントが隠されてしまうことを防ぐために，ドラッグ中のコンポーネントは指の少し上方に表示する．

コンポーネントの機能の実行

コンポーネントメニューの Exe を選択すると，コンポーネントメニューを開いたコンポーネントの機能が実行される．コンポーネントに出力がある場合は，その出力がダイアログに表示される．

3.6.3 セルに対する操作

セル上のコンポーネントが表示されてない位置を長押しすることにより，セルメニューを表示することができる．図 3.14 にセルメニューを示す．セルメニューの項目を選択することによって，セルの移動，セルの削除の操作を実行することができる．

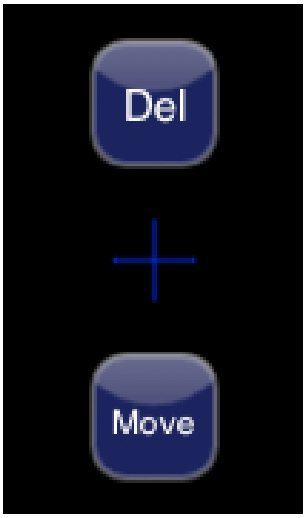

図 3.14: セルメニュー

セルの移動

セルメニューから,「Move」を選択すると，セルをドラッグできるようになり，移動したいセルとセルの間にドロップすると，セルの順番を変更することができる．この際，ドラッグ

をしているセルがコンテキスト定義のセルであり，かつ，隣接するセルとの関係が OR である場合は，隣接する OR 関係のセルも同時に移動される．

セルの削除

セルメニューから「Del」を選択すると，長押ししたセルが削除される．この際，削除したセルがコンテキスト定義のセルであり，かつ，隣接するセルとの関係に OR がある場合は，セルを削除したことにより，隣接することになったセル間の関係を隣接する前のセル間の関係を保持したまま削除を行う．

3.6.4 プログラムの保存

メニュー画面の保存ボタンをタップすることにより，保存を行う．コンテキスト定義，アクション定義にブランクコンポーネントが含まれる場合は，プログラムが作成途中であるため，警告が表示され，保存されない．

3.7 プログラムの管理機能

プログラム管理機能は，過去に作成したプログラムの閲覧や編集，削除，コピーを行うための機能である．

3.7.1 画面構成

図 3.15 にプログラム管理画面の構成を示す．プログラムのコンテキスト定義が画面上方に，アクション定義が下方に表示される．仕切り線中央の矢印を上下にドラッグすることで，コンテキスト定義とアクション定義の大きさを変更することができる．プログラムが複数ある場合は，画面の左右の端に他のプログラムがあることを示す図 3.3 と同様の影が表示されるので，外側から内側に向けてクロッシング操作を行うことで，前後のプログラムに表示が切り替わる．画面の上部には，メインメニュー画面に戻るボタンが配置され，画面下部には，編集ボタン，削除ボタン，コピーボタンが配置されている．

3.7.2 プログラムの編集

編集ボタンをタップすると，現在表示しているプログラムの編集画面に切り替わる．編集画面は，プログラム作成画面と同様である．コンテキスト定義やアクション定義を編集し，選択したプログラムに上書き保存する．

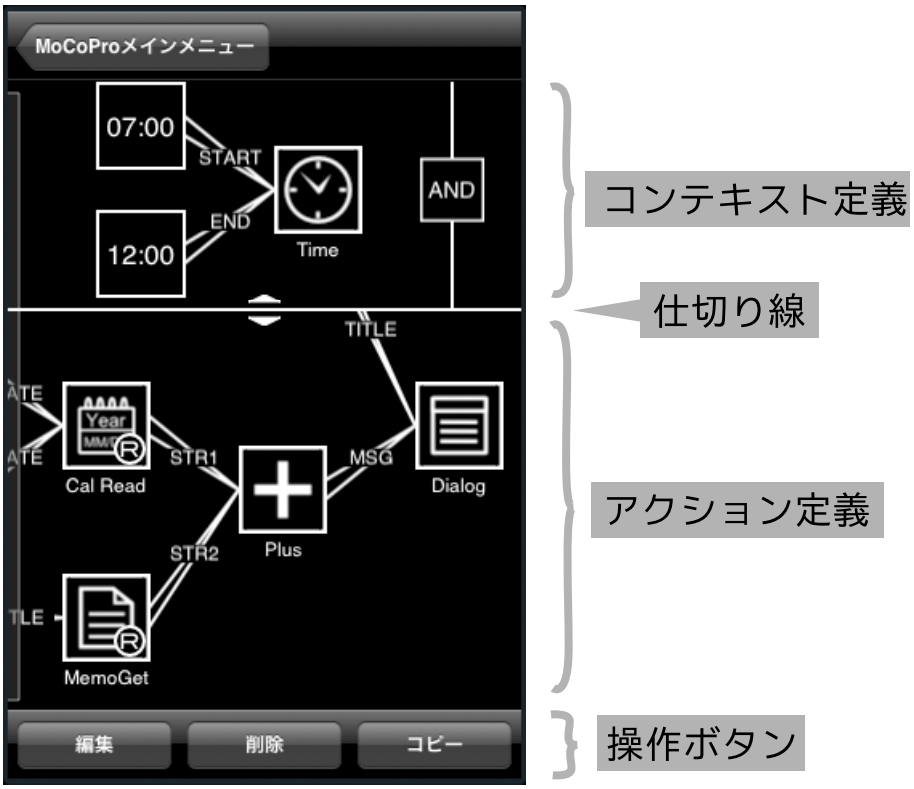

図 3.15: プログラム管理画面の構成

3.7.3　プログラムのコピー

コピーボタンをタップすると，現在表示しているプログラムを複製する．

3.7.4　プログラムの削除

削除ボタンをタップすると，現在表示しているプログラムを削除する．

3.8　プログラムの実行機能

プログラムの実行機能は，コンテキスト依存プログラムのコンテキスト定義と現在のコンテキストを比較し，コンテキスト定義に示す条件を満たしている場合は，アクション定義に示される処理を実行する．

3.8.1　画面構成

プログラムの実行機能は，待ち受け画面を表示することにより実行が開始される．待ち受け画面の構成を図 3.16 に示す．画面には，プログラムの実行機能によって得られたコンテキストや実行したプログラムのログが残される．メインメニューに戻るボタンをタップすると，メインメニュー画面に戻り，コンテキストの取得が停止される．

3.8.2　コンテキストの取得とプログラムの実行

待ち受け画面を表示すると，コンテキストの取得が開始される．コンテキストが取得されるたびに，取得した時刻，コンテキストの種類，コンテキストの状態が 1 組として画面にログが残される．取得したコンテキストが，プログラムのコンテキスト定義の条件を満たすと，プログラムのアクション定義に従って，処理が実行される．この際，実行された時間とプログラム ID がログとして画面に残される．複数のプログラムが同時にコンテキスト定義の条件を満たした場合は，最初に作成されたプログラムから順に実行される．

図 3.16: 待ち受け画面の構成

第4章　MoCoProの実装

本章では，MoCoPro の実装の詳細について述べる．MoCoPro は，iPhone SDK 2.2 を用いて開発を行い，iPhone，iPod touch 上で動作するネイティブアプリケーションとして実装した．プログラミング言語は，Objective-C 2.0 を用いている．

4.1　システム構成

システム構成を図 4.1 に示す．MoCoPro は，コンテキスト解釈部，コンテキスト管理部，アクション実行部，ビジュアルエディタ部，定義データベースからなる．MoCoPro は，入力としてコンテキスト源からコンテキストの変化を受け取り，出力として携帯端末やネットワーク上の機能を実行する．

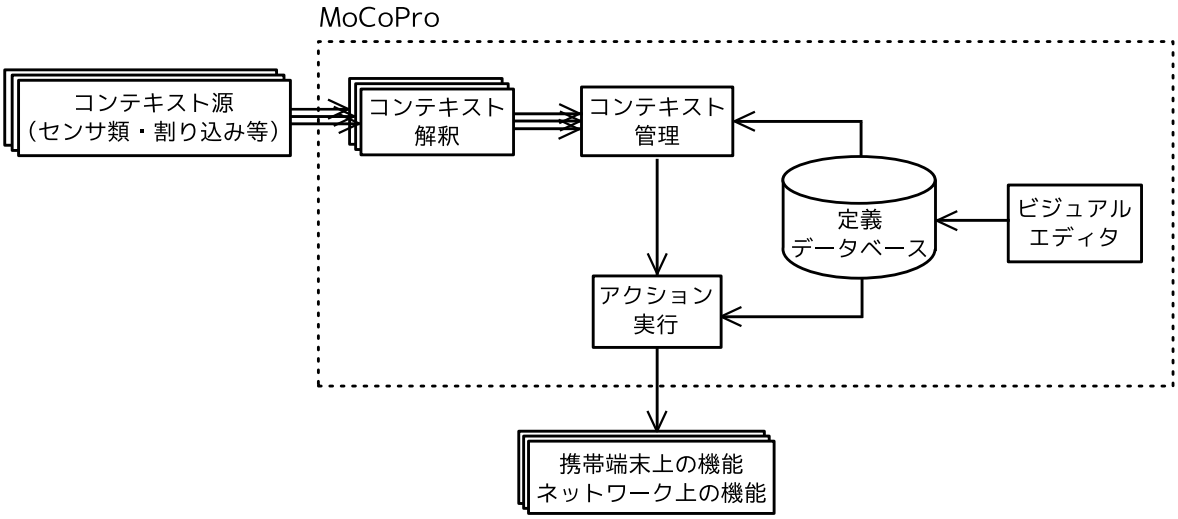

図 4.1: システム構成

コンテキスト源

コンテキスト源は，携帯端末の機能やセンサであり，コンテキスト源の出力は機能やセンサの出力そのものである．コンテキスト源は，それぞれ独立して動作しており，非同期にデータを出力する．例を挙げると，時計から得られる日時やメール受信による割り込み，GPS センサから得られる緯度経度や加速度センサから得られる 3 軸方向の加速度値などである．

コンテキスト解釈部

コンテキスト解釈部は，入力としてコンテキスト源からデータを受け取り，それを内部形式データであるコンテキストデータへと変換し出力する．

コンテキストデータのメンバを表 4.1 に示す．コンテキストデータは，コンテキストの種類とコンテキストの値の 2 つのメンバを持つ．例えば，場所の場合は，GPS センサから緯度経度を受け取る．場所のコンテキスト解釈は，コンテキストの種類を LATITUDE（緯度）とし，コンテキストの値を GPS センサから得られた緯度値とするコンテキストデータと，コンテキストの種類を LONGITUDE（経度）とし，コンテキストの値を GPS センサから得られた経度値とするコンテキストデータの 2 つのコンテキストデータを出力する．また，メール受信の場合は，機能からメール受信の割り込みの情報を受け取る．コンテキストの種類を MAILRECEIVE（メール受信）とし，コンテキストの値を受信したメールの送信者のメールアドレスとするコンテキストデータを出力する．

また，同一のコンテキスト源からデータを受け取った場合でも，解釈の違いによって，異なるコンテキストとして扱うことが可能である．この解釈は，専門家が行い，ユーザはコンテキストコンポーネントを使用することで，様々な解釈を行ったコンテキストの条件を記述することができる．例えば，加速度センサの場合は，3 軸方向の加速度値を出力するが，解釈の仕方を変えることにより，ユーザの動作や携帯端末の方向を表すコンテキストとして解釈することができる．ユーザの動作のコンテキストを得る場合は，連続的な振動を認識し，振動の大きさにより,「走っている」,「歩いている」,「止まっている」の動作とする．携帯端末の方向を得る場合は，特定の方向に一定強度の加速度値が一定時間持続していることを検出することにより表向き，裏向きなどの携帯端末の方向とすることができる．このような場合は，1 つのコンテキスト源に対して，複数のコンテキスト解釈を割り当てる．

表 4.1: コンテキストデータのメンバ

メンバ名	型	説明
type	NSString	コンテキストの種類
value	コンテキストを表すことに適した型	コンテキストの状態

コンテキスト管理部

コンテキスト管理部は，入力としてコンテキストデータを受けとり，それを定義データベースに保存されているコンテキスト定義と比較し，コンテキスト定義の条件を満たしている場合は，そのコンテキスト定義を含むプログラムのプログラム ID を出力する．

まず，定義データベースから読み出したコンテキスト定義をコンテキストデータと比較可能な形式であるコンテキスト定義データに変換しておく．コンテキスト定義データのメンバを表 4.2 に示す．また，比較用演算子の取りうる値を表 4.3 に示す．メンバの値は，コンテキスト定義の記述に従って決定される．そして，入力として，コンテキストデータを受け取ると，コンテキストデータとコンテキスト定義の条件との比較を行う．この際，コンテキストデータとコンテキスト定義データのコンテキストの種類が同一のもののみ比較を行う．比較は，コンテキスト定義データの比較用演算子に示す方法で行う．比較の結果，コンテキスト定義の条件を満たしている場合は，そのコンテキスト定義を含むプログラムのプログラム ID を出力する．

表 4.2: コンテキスト定義データのメンバ

メンバ名	型	説明
type	NSString	コンテキストの種類
operator	OperatorType	比較用演算子
value1	コンテキストを表すことに適した型	コンテキストの状態 1
value2	コンテキストを表すことに適した型	コンテキストの状態 2

表 4.3: 値の比較用演算子

列挙子	説明
EQUAL	値 1 = 比較値
NOT_EQUAL	値 1 $\neq$ 比較値
GREATER	値 1 $<$ 比較値
LESS	値 1 $>$ 比較値
GREATER_EQUAL	値 1 $\leq$ 比較値
LESS_EQUAL	値 1 $\leq$ 比較値
IN_RANGE	値 1 $\leq$ 比較値 $\leq$ 値 2

アクション実行部

アクション実行部では，入力としてプログラム ID を受け取り，出力としてその携帯端末やネットワーク上の機能を実行する．

携帯端末やネットワーク上の機能の実行は，プログラム ID で示されるプログラムのアクション定義に従って行われる．処理の実行は，アクション記述のデータフロー図のルートとなる出力のないコンポーネントの機能を実行することから始まり，入力の値が必要となった際に，入力に接続されているコンポーネントを実行し，その出力を得ることによって処理が進められる．

ビジュアルエディタ部

ビジュアルエディタ部は，ユーザにコンテキスト定義とアクション定義を記述するためのインタフェースを提供する．

ビジュアルエディタ部のインタフェースや機能については，第 3 章に示した通りである．ユーザがプログラムを記述し，プログラムの保存を実行すると，定義データベースにプログラムが保存される．

定義データベース部

データベース部は，コンテキスト定義とアクション定義を保持し，ユーザの要求に応じて定義の出力や編集，削除を行う．また，ユーザが作成した関数の保持も同様に行う．データベースには，SQLite を使用している．

4.2 コンポーネントの実装

MoCoPro では，新たなコンポーネントを追加することで，新たな機能を利用するプログラムを作成できるようになる．以下では，コンポーネントの実装方法について述べる．追加可能なコンポーネントは，オペレーションコンポーネント，コンテキストコンポーネントである．オペレーションコンポーネントとコンテキストコンポーネントは，同一のクラスである ActionComponent クラスを継承し，メンバの初期化を行う initialize メソッドと，機能の実行を行う execute メソッドを実装する．初期化を行う必要があるメンバを表 4.4 に示す．

例として，図 4.2 に Plus コンポーネントの initialize メソッドの実装を示す．

表 4.4: コンポーネントクラスのメンバ

メンバ名	型	説明
icon	NSString	アイコンの画像ファイル名
title	NSString	コンポーネントのタイトル
description	NSString	コンポーネントの説明
outType	NSString	出力型
inTypes	NSArray (NSString クラスの配列)	入力型の配列
inInfos	NSArray (NSString クラスの配列)	入力の説明の配列
componentType	ComponentType	コンテキスト定義とアクション定義 のどちらで使用できるコンポーネントか
choices	NSArray	入力型が選択型である際の選択肢

```
- (void)initialize
{
  icon = @"plus.png";
  title = @"Plus";
  description = @"入力した 2 つの文字列を連結し出力する";
  outType = @"STRING";
  inTypes = [[NSMutableArray alloc]
                          initWithObjects:@"STRING", @"STRING", nil];
  inInfos = [[NSMutableArray alloc]
                          initWithObjects:@"STR1", @"STR2", nil];
  componentType = BOTH;
}
```

図 4.2: Plus コンポーネントのメンバの初期化コード

次に，コンポーネントの機能の実装である，execute メソッドについて述べる．execute メソッドでは，入力に何らかの計算を施し，その結果を出力する．この出力は，出力先のコンポーネントの入力となる．図 4.3 に，Plus コンポーネントの execute メソッドの実装を示す．

```
- (void)execute
{
  // STRING 型の入力を 2 つ受け取る（returnValues 配列に入力の値が保存される）
   NSString *str1 = [returnValues objectAtIndex:0];
   NSString *str2 = [returnValues objectAtIndex:1];

  // 処理の結果を出力する
   [self outputValue:[str1 stringByAppendingString:str2]];
}
```

図 4.3: Plus コンポーネントの実行コード

初期化すべきメンバや，実装するメソッドについては，オペレーションメソッドとコンテキストメソッドに違いはないが，execute メソッドの返り値が異なる．

コンテキストコンポーネントが実行された際は，コンテキスト定義データを出力する．図 4.4 に，Locatoin コンポーネントの場合の execute メソッドの実装を示す．

```
- (void)execute
{
  // LOCATION 型の入力を 2 つ受け取る（returnValues 配列に入力の値が保存される）
  Location *loc = [returnValues objectAtIndex:0];

  // 緯度のコンテキスト定義データを作成する
  ContextDefinitionData *latitude =
                            [[ContextDefinitionData alloc] init];
  latitude.type = @"LATITUDE";
  latitude.value1 = loc.latitude - loc.range.latitude;
  latitude.value2 = loc.latitude + loc.range.latitude;
  latitude.operator = IN_RANGE;

  // 経度のコンテキスト定義データを作成する
  ContextDefinitionData *longitude =
                            [[ContextDefinitionData alloc] init];
  longitude.type = @"LONGITUDE";
  longitude.value1 = loc.longitude - loc.range.longitude;
  longitude.value2 = loc.longitude - loc.range.longitude;
  latitude.operator = IN_RANGE;

  // コンテキスト定義データの配列を出力する
  [self outputValue:
              [NSArray arrayWithObjects:latitude, longitude, nil]];
}
```

図 4.4: コンテキストコンポーネントの実行コード

4.3 コンポーネントの入出力型

コンポーネントの入出力型は，コンポーネントのメンバである inTypes や outType で示される文字列によって識別される．この型によって，接続可能なコンポーネントやドラッグアンドドロップが可能なコンポーネントが制限される．現在の実装で使用されている型を表 4.5 に示す．

コンポーネントを実装する専門家は，inputTypes や outType に新たな文字列を選択することで，新しい型を作成することができる．例えば，郵便番号型を出力するコンポーネントを作

表 4.5: コンポーネントの型の種類

型の名称	型の説明	データを表すクラス
STRING	文字列型	NSString
NUMBER	実数値型	double
BOOLEAN	ブール値型	BOOL
LOCATION	場所型	Location
DATETIME	日時型	NSDate
CHOICE	選択型	NSString
LIST	配列型	NSArray
NONE	出力なし	-

成する場合は，outType を ZIPCODE や POSTCODE などにすることにより，新たな型を作成することができる．実際に出力するデータは，同一の型を持つコンポーネントで統一されていれば，NSString や NSValue など，どのようなクラスのオブジェクトを使用しても良い．

4.4 クロッシング操作の実装

クロッシング操作の実装について述べる．MoCoPro におけるクロッシング操作は，携帯端末の画面の端を指で横切ることにより，画面の切り替えを行う操作である．図 4.5 の矢印が示すように，携帯端末の画面の外側から内側に向かって指を滑らす．携帯端末の画面の外側はタッチパネルではないため，指の動きを検出できないが，指が画面に入ると同時にタッチパネルにより，指の動きを検出することができる．クロッシング操作が行える画面の辺には，図 4.5 の赤色の矩形領域が示すように，クロッシング操作を検出するためのビューを配置しており，このビューを特定の方向に，一定以上の距離を，一定時間内で指を横切ることにより，クロッシング操作の実行を行う．現在の実装では，実際に iPhone 上で操作を行った際に，操作が行いやすかった，距離を 40px，時間を 0.1sec 以内としている．方向や距離，速度の制限を設けたのは，画面をスクロールする際や，コンポーネントセレクタの操作を行っている際に，誤ってクロッシング操作が実行されないようにするためである．

4.5 自動配置

システムによって自動的に行われるコンポーネントの配置について述べる．図 4.6 にコンポーネントの配置に関する各部の名称を示す．

まずコンポーネントの配置は，をデータフロー図のルートとなるコンポーネントをセルの中央に配置することから行う．このコンポーネントをルートコンポーネントと呼ぶ．次に，ルートコンポーネントの子コンポーネントの配置を式 4.1，4.2，4.3 に従って行う．これをルートコンポーネントから末端に向かって再帰的に計算することによりコンポーネントの配置を行う．

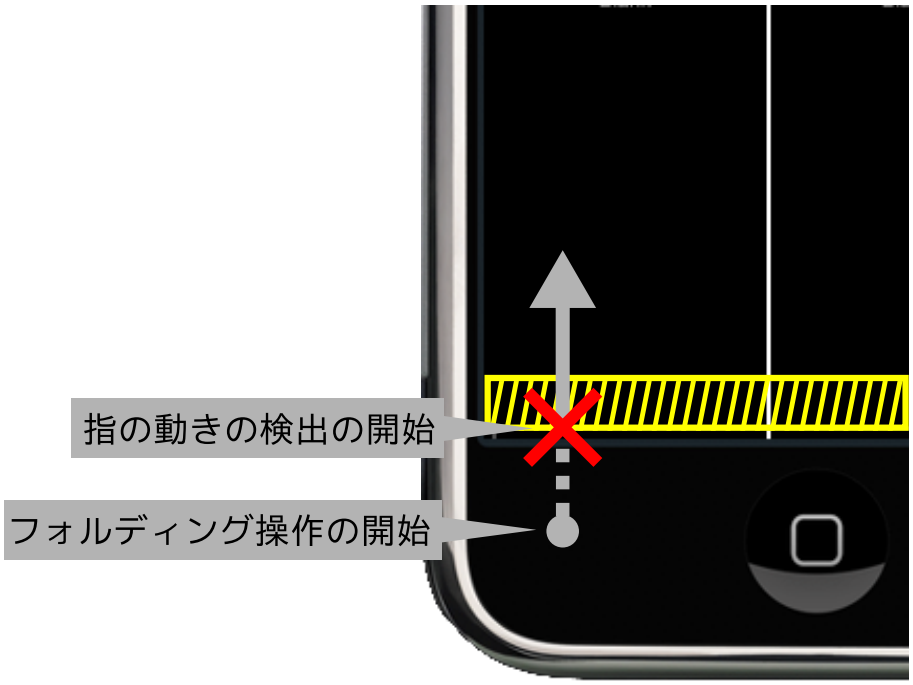

図 4.5: クロッシング操作の検出方法

ルートコンポーネントは，画面の縦幅の中央になるように配置する．他のセルのルートコンポーネントの高さも同じになるように配置することで，ルートコンポーネントの位置を分かりやすくする．また，コンポーネントの間隔は常に，X 軸方向には $spanX$，Y 軸方向には $spanY$ となるように配置する．

フォルディングされたコンポーネントは，フォルディングを開始したコンポーネントと少しずらした場所に配置され，コンポーネントの $width$ や $height$ は 0 として計算される．フォルディングされたコンポーネントの子コンポーネントは，フォルディングを開始したコンポーネントの子コンポーネントと同様に計算され，配置される．

$$positionX_i = positionX_{i-1} - COMPONENT\_WIDTH - spanX \tag{4.1}$$

$$positionY_i = \left(positionY_{i-1} - \frac{height - height_i}{2} \right) + \sum_{j=0}^{i} height_j \tag{4.2}$$

$$height = \begin{cases} COMPONENT\_HEIHGT & (\mathrm{n} = 0) \\ \sum_{i=1}^{n} height_i + spanY * (n-1) & (n > 0) \end{cases} \tag{4.3}$$

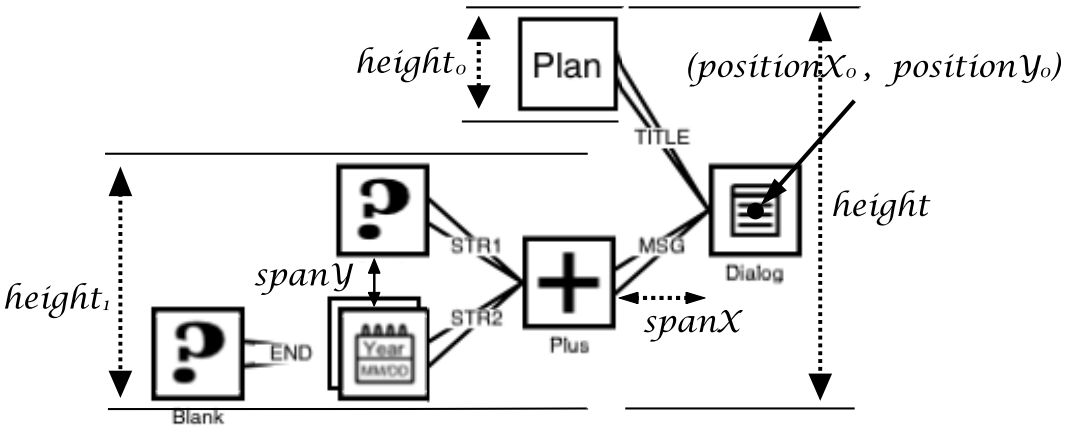

図 4.6: コンポーネントの配置

第5章　利用シナリオ

本章では，MoCoPro の利用シナリオについて述べる．まず，MoCoPro で使用できるコンポーネントの例を示し，次に，MoCoPro が有効に活用できる利用シーンについて述べる．

5.1　コンポーネント例

図 5.1 に MoCoPro で使用できるコンポーネントの一部を示す．4.2 章に示す手順に従えば，ユーザが新たな機能を持つコンポーネントを追加することができる．

図 5.1: 待ち受け画面の構成

5.2　利用シーン 1

A 君は，買いたい本があるにもかかわらず，本屋の近くを通りかかっても，いつも立ち寄るのを忘れてしまう．そこで，MoCoPro を使用して,「本屋の付近にいるときは，ブザーとともに，買いたい物のリストを表示する」というプログラムを記述した．図 5.2 から図 5.9 は，プログラム完成までの一連の操作を示している．

作成したプログラムはうまく動作していたが，買いたい本のリストに何も書かれていない時においても表示されることや，勤務時間内であっても表示されることを煩わしく感じていたので，買いたいものリストに項目があることと，勤務時間外であることを条件として追加した．また，友人に次に本屋に寄ったときには，買ってきて欲しい本があると言われたので，リマインダとして友人に電話をかける処理を追加しておいた．以上の条件と処理の記述は，プログラム管理画面で，作成したプログラムを表示し，編集ボタンを押すことで，編集を行った．

図 5.10 は，完成したプログラムの全体像を示している．コンテキスト記述には，本屋の付近であること，時刻が 12:00-13:00 もしくは 18:00-23:00 であること,「Book Wish List」というメモが空でないこと，の 3 つの条件が記述されている．アクション記述には，ブザーを鳴らすこと,「Book Wish List」というメモをダイアログで表示すること,「090-1234-5678」に電話をかけることの 3 つの処理が記述されている．

仕事を終えて帰宅していた A 君は，本屋の近くを通りかかったとき，携帯端末のバイブレーションが起動したので見てみると，図 5.11 のように買いたい本のリストが表示されていた．買いたい本を確認し，ダイアログの OK ボタンをタップすると，電話をかける処理の開始を示すダイアログが図 4.4 のように表示されたので，友人に電話をかけ，忘れないように用件を聞くことができた．

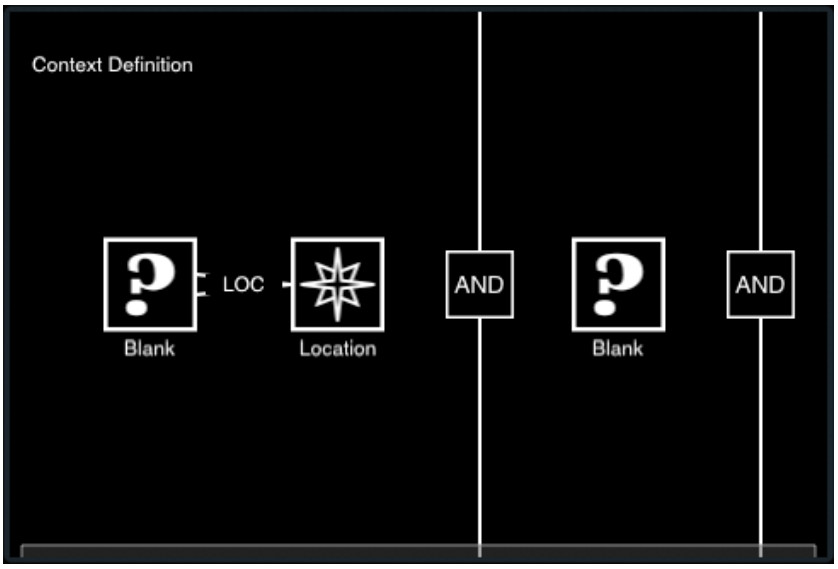

図 5.2: 場所のコンテキストを監視するコンポーネントの配置

図 5.3: 地図上の領域を選択しデータを作成

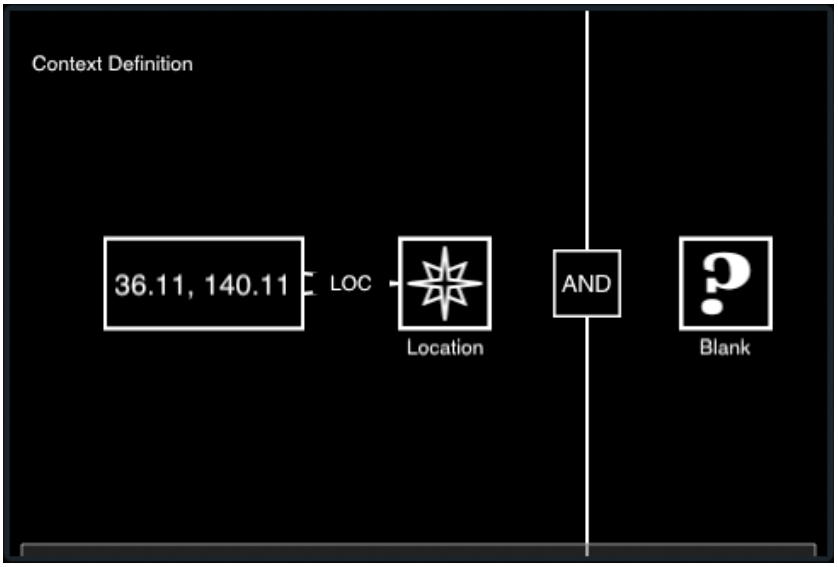

図 5.4: 本屋の付近という条件を示すコンテキスト記述

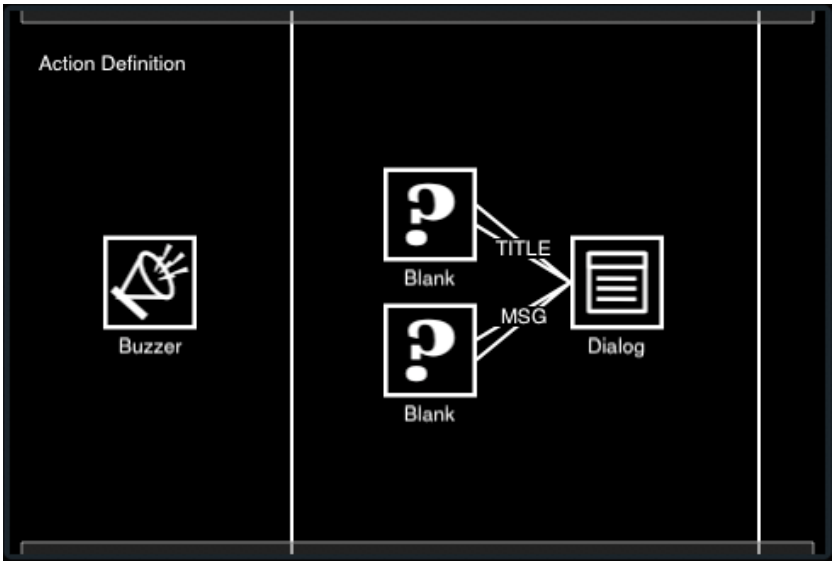

図 5.5: ブザーコンポーネントとダイアログコンポーネントの配置

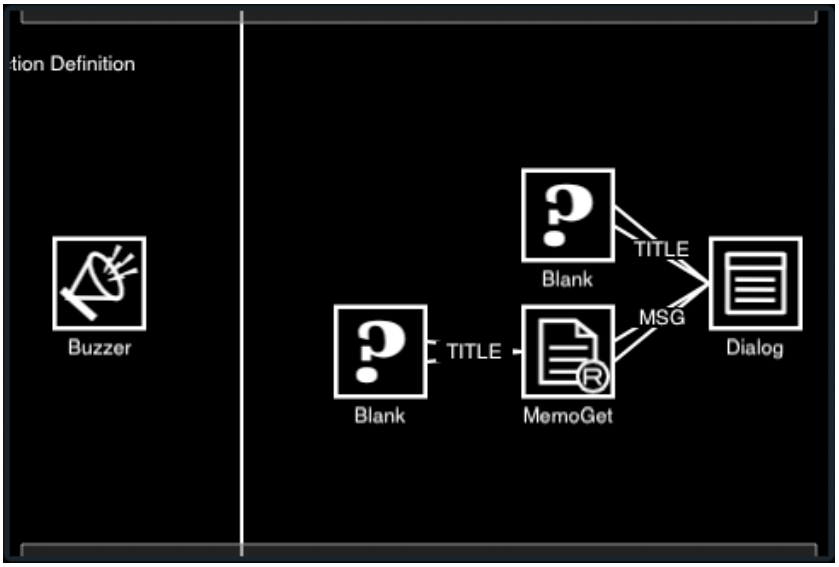

図 5.6: メモ取得コンポーネントの配置

図 5.7: ダイアログのタイトルと読み込むメモのタイトルを手入力

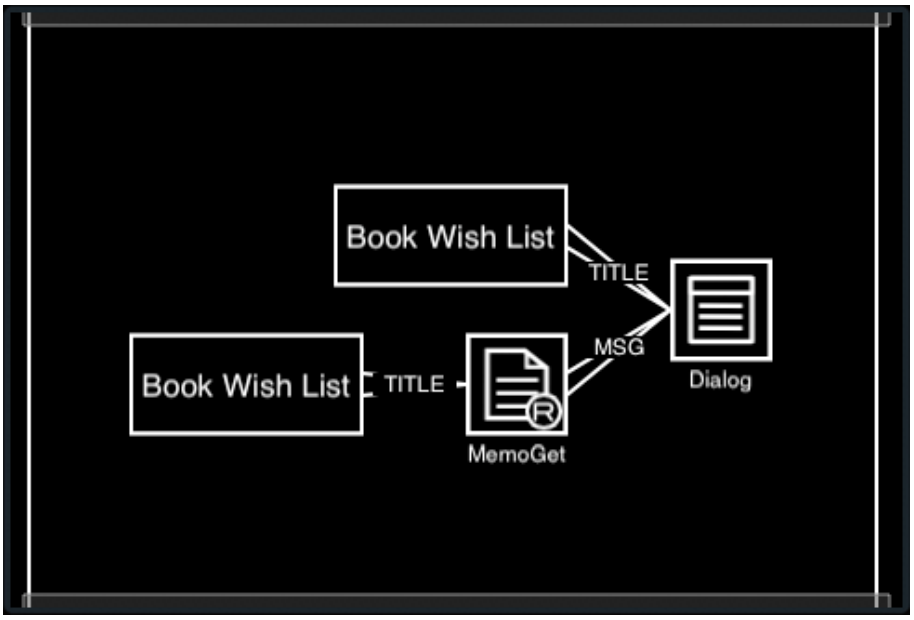

図 5.8: 買いたい本のリストを表示する処理を行うアクション記述の完成

図 5.9: プログラムを保存

図 5.10: 完成したプログラムの全体像

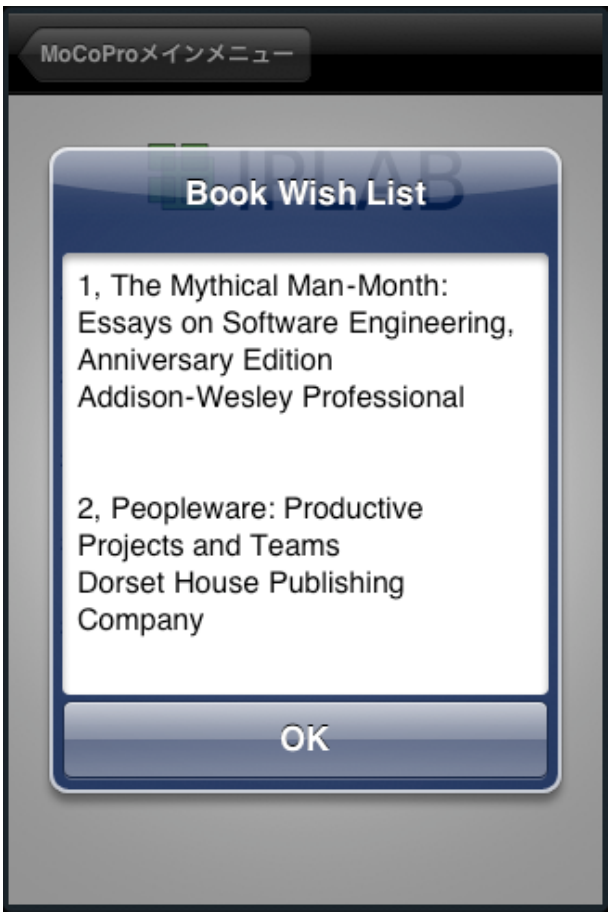

図 5.11: 買いたい本リストの表示

図 5.12: 通話開始の確認を行うダイアログ

5.3　利用シーン 2

B さんは，会社からの帰宅する際，駅に到着したときに，家族にもうすぐ家に帰ることをメールで伝えている．毎日のように行うタスクであるので，駅に到着した際に自動的にメールを送信できれば便利だと考えた．また，最近働き過ぎであるように感じていたので，スケジュール帳に帰宅時間を記録していこうと考えた．そこで，MoCoPro を利用して,「夜帰宅する際，駅に到着したときに，家族にメールを送る．また帰宅時間をスケジュール帳に記録する」というプログラムを記述した．図 5.13 から図 5.22 は，プログラム完成までの一連の操作を示している．

図 5.16 は，帰宅時間をスケジュール帳に記録するアクション記述を示している．スケジュール登録コンポーネントは，スケジュールを登録する日時と，スケジュールの内容を入力として受け取り，スケジュール帳にスケジュールの登録を行う．これを利用して，プログラムの実行時の日時に,「Return Home」という内容のスケジュールを登録し，帰宅時間の記録を行っている．図 5.22 は，家族にメールを送信する処理を行うアクション記述を示している．ループコントロールコンポーネントにメールアドレスのリストを入力し，リストのメールアドレスのそれぞれに対して題名がなく，本文が「I'm coming Home」というメールを送信する処理を行う．完成したプログラムの全体像を図 5.23 に示す．

完成したプログラムにより，駅に到着すると自動的にメールを送信し，スケジュール帳に帰宅時間を記録するようになった．

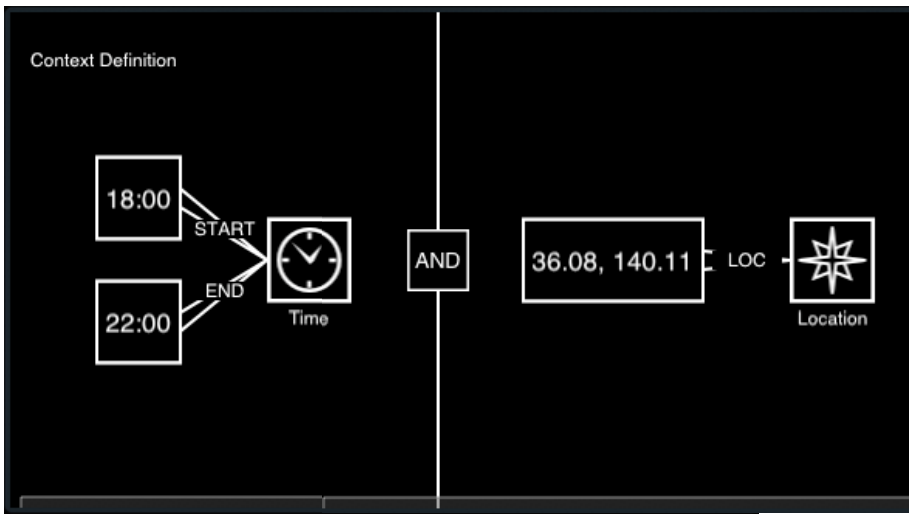

図 5.13: 「18:00-22:00 の間に駅にいる」というコンテキスト定義

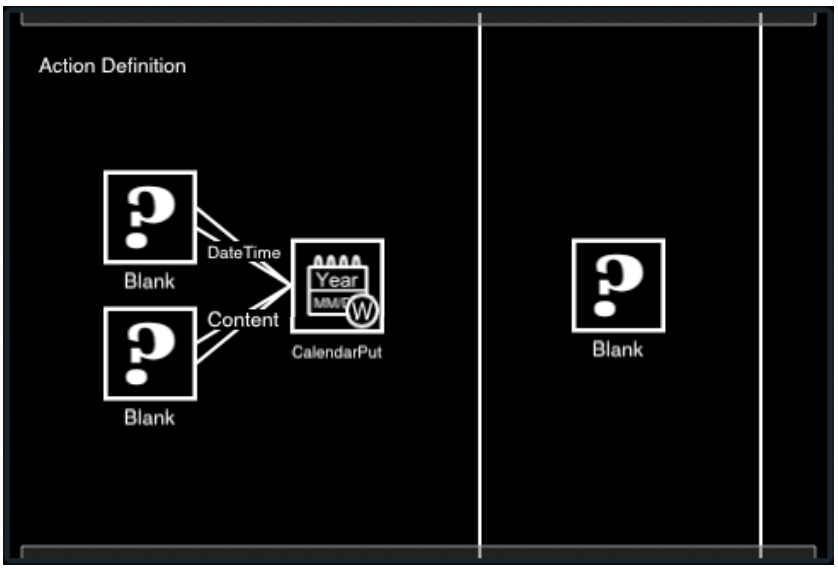

図 5.14: スケジュール登録コンポーネントの配置

図 5.15: 登録するスケジュールの日時を手入力

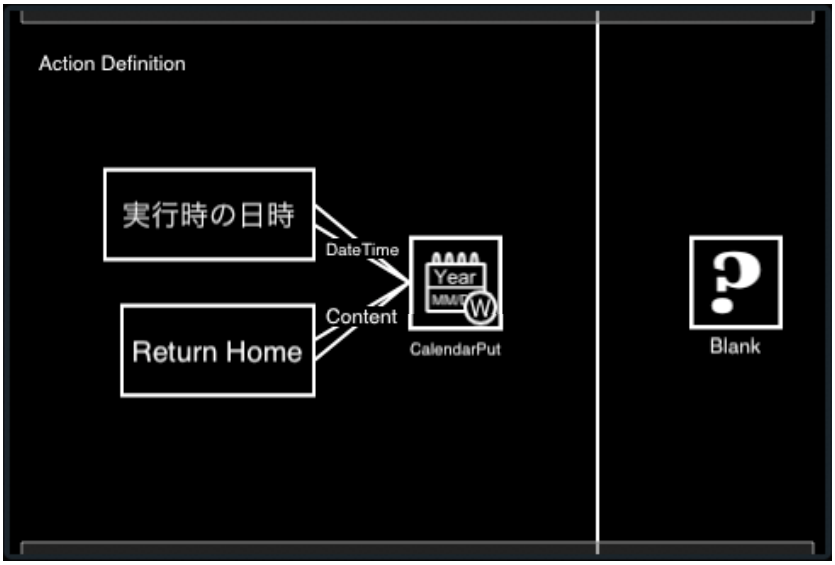

図 5.16: 帰宅時間をスケジュール帳に登録する処理をするアクション記述

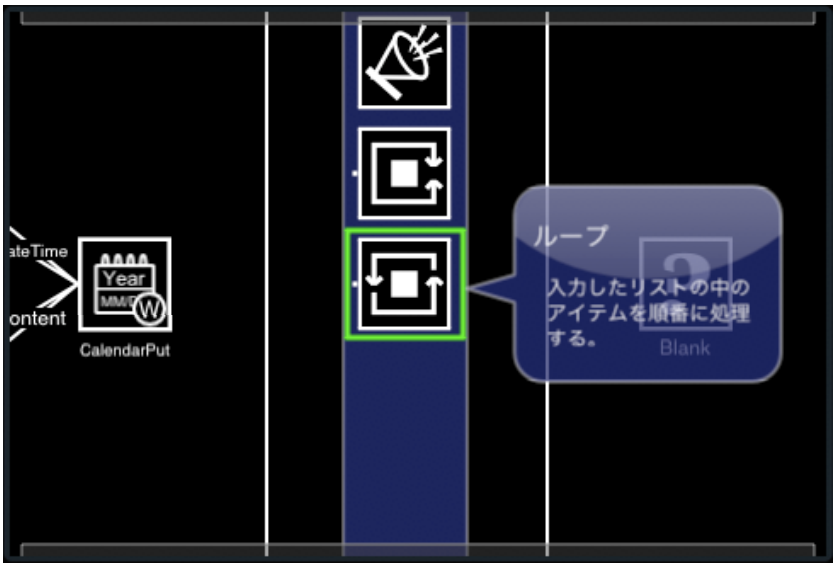

図 5.17: ループコントロールコンポーネントを選択する

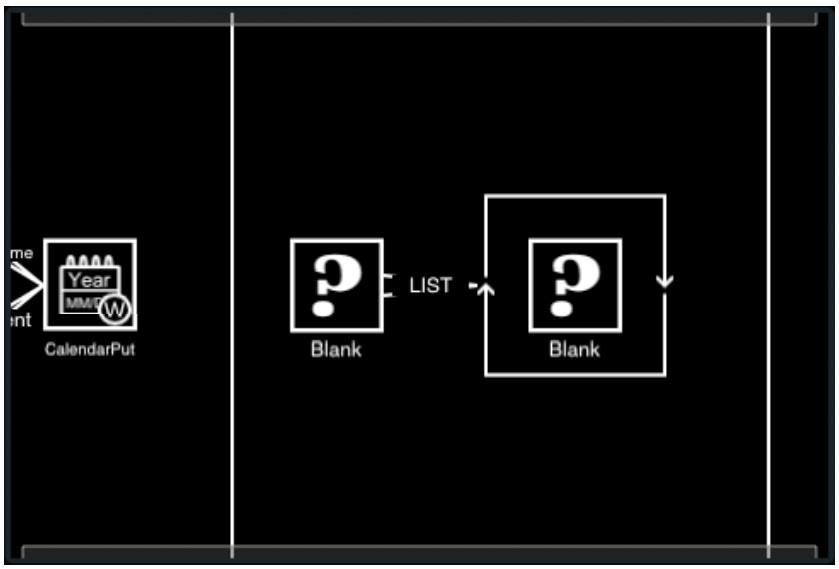

図 5.18: ループコントロールコンポーネントの配置

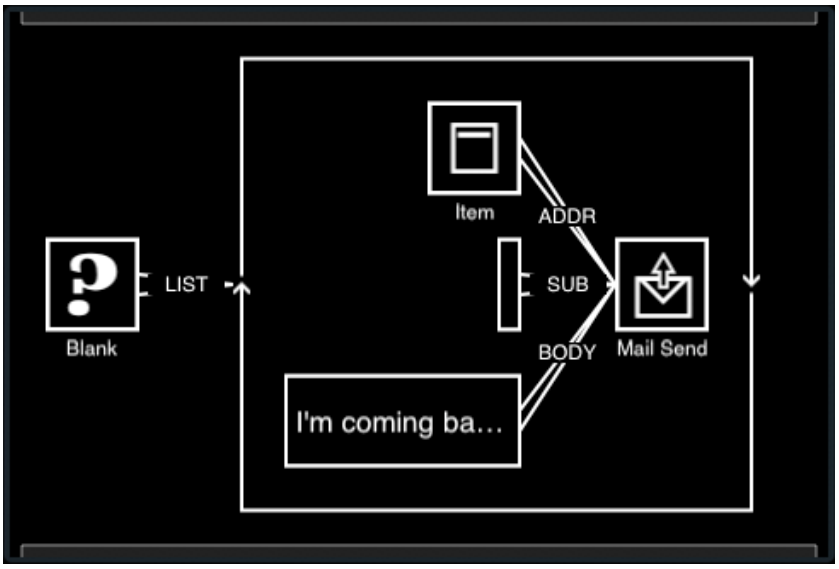

図 5.19: メールの送信先，題名，本文を指定

図 5.20: ループコントロールコンポーネントの入力となるリストの要素数を手入力

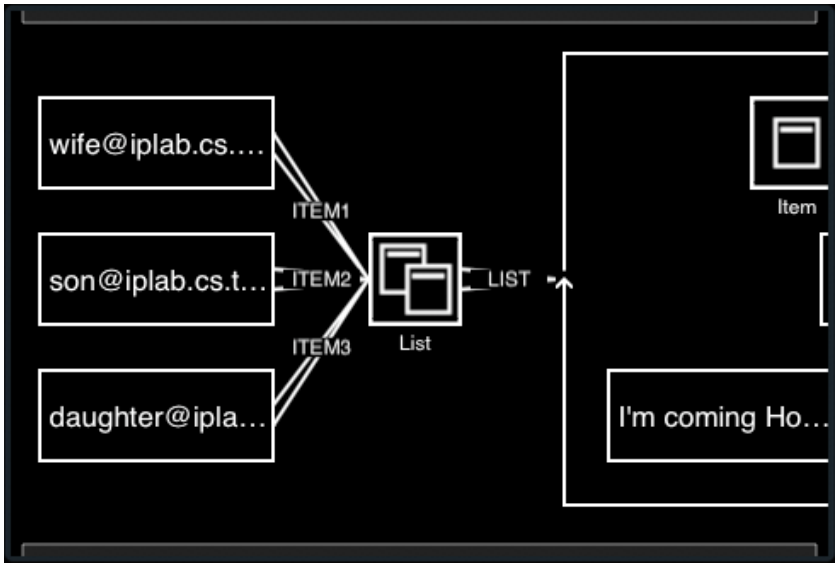

図 5.21: ループコントロールコンポーネントへ入力するリストの要素を入力

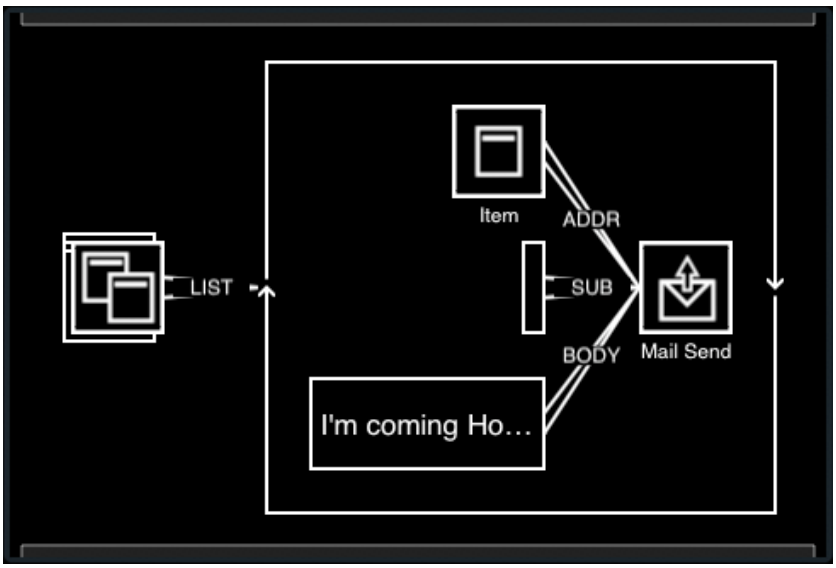

図 5.22: 家族にメールを送信する処理を行うアクション記述

図 5.23: 完成したプログラムの全体像

第6章　関連研究

本研究が対象としている研究分野と関連する研究について述べる．まず，コンテキストアウェアシステムの開発を支援する手法や環境に関する研究を本研究との比較と共に述べる．次に，携帯端末上のタッチパネルの操作に関する研究を本研究との関連と共に述べる．

6.1　コンテキストアウェアシステムの開発支援に関する研究

データフロー図やビジュアルプログラミング言語を用いることによって，コンテキストアウェアシステムの開発を支援する研究が行われている．Bannach らの CRN toolbox[7] は，ユーザの行動やコンテキストを認識するシステムの構築を支援する環境である．複数のセンサを用いるような分散処理に適しており，様々なアルゴリズムを実装したコンポーネントを接続し，データフロー図を描くことで，センサデータの加工や，センサ間の同期を取ることが可能である．Weis らの VisualRDK[23] は，パーベイシブアプリケーションをプロトタイピングするための，専門家やホビープログラマを対象としたビジュアルプログラミング言語である．人物や場所などから得られるイベントに反応して，ハードウェアやソフトウェアが行う処理のプロセスを，様々な機能を持つコンポーネントや制御構文を用いて記述することができる．テキスト形式のプログラミング言語ではなく，データフロー図やビジュアルプログラミング言語を用いる点が本研究と同様である．しかし，MoCoPro が携帯端末上で利用する環境であることに対して，CRN Toolbox や VisualRDK は，PC 上で利用する環境である．また，CRN Toolbox は，センサデータを加工し，コンテキストを得ることを目的としているが，MoCoPro は，ユーザの望んだコンテキストの条件を定義することを目的としている点においても異なる．VisualRDK では，コンテキストの条件とバードウェアやソフトウェアが行う処理を混在させてプロセスを記述するが，MoCoPro では，コンテキストの条件と処理を分割して記述することで，条件や処理の追加や削除，組み合わせの変更などの管理を行いやすくすることができる．

ユーザが行ったタスクを，コンピュータに認識させることにより，半自動的にコンテキストアウェアシステムを構築する研究が行われている．Anind らの a CAPpella[10] は，PbD（Programming by Demonstration）を採用した環境である．ユーザがタスクを行っている際のセンサデータやイベントをコンピュータが認識し推論した処理を，ユーザが繰り返し訂正していくことによって，半自動的にタスクを補助するシステムを構築することができる．加藤らのシステム [28] では，コンテキストを扱うシステムの多くが，コンテキストの状態を条件として何らかの動作を行うものであることに着目し，アプリケーションの実行履歴から，実

行時のコンテキストを汎化することで，曖昧なコンテキストを条件としてアプリケーションやサービスを実行するプログラムを作成することを可能にしている．これらの手法の最大の利点は，プログラムを記述する必要がないため，エンドユーザにとって扱いやすいことである．MoCoPro では，プログラムを記述する必要があるため，ユーザに多少のプログラムの知識を求めてしまうが，ビジュアルプログラミング形式や簡易な記述形式を採用することにより，エンドユーザであってもプログラムを記述できるようにすることを目指している．

携帯端末上で，コンテキストアウェアシステムを開発する環境の研究について述べる．Korpipaa らの Context Studio[13] は，メニュー形式のインタフェースからコンテキストの条件や処理を選択することにより，コンテキストアウェアシステムを開発するシステムである．Kollet らの PDAGraph[5][12] は，Prograph[8] を携帯端末上に移植し，イベントを扱えるようにしたビジュアルプログラミング言語である．ユーザインタフェースを扱う UI コンポーネント，外部モジュールを扱う External コンポーネント，既存のスクリプトやユーザが作成したスクリプトである Script の 3 つのコンポーネントを組み合わせることにより，プログラムを記述していく．寺田らの Wearable Toolkit[27] は，ウェアラブルコンピューティング環境でプログラミングを行うためのプラットフォームである．Wearable Toolkit では，ECA ルールに基づいてコンテキストの条件や処理を定義する．また，コンテキストを定義するための環境もあり，過去のセンサデータの平均値や変化量などの特徴量をコンテキストとして登録することができる．Context Studio は，メニュー形式を用いるため，キーボタンを中心とした携帯端末であっても操作が行いやすいことに加え，メニューから選択することだけでプログラムを記述できるという利点がある．しかし，MoCoPro では，コンテキストの条件間の関係や処理の実行方法，機能の組み合わせなど，ユーザの多種多様な要求に応えるためより複雑な動作な条件や処理を記述することができる．PDAGraph は，MoCoPro と多くの部分で類似するが，プログラムの記述方式と画面構成の点で異なる．PDAGraph は，イベントに反応するプログラムを記述することができるが，コンテキストの条件と処理の記述をどのように分割して，記述するのかはユーザに任されている．また，プログラムを複数の画面に分割して記述するため，プログラムの流れが分かりづらい．MoCoPro では，コンテキストの条件と処理の実行を分割して記述するため，プログラムの作成や管理が行いやすい．また，MoCoPro では，定義の記述を分割することやフォルディングなどの操作を用いることで，一画面内で定義を記述することを可能にし，プログラムの流れの把握を行いやすくしている．Wearable Toolkit は，MoCoPro と同様に ECA ルールに基づいて記述を行う．エンドユーザ向けのツールとして，コンテキストを定義するツールや GUI により ECA ルールを記述する環境があるが，MoCoPro で記述できるような様々な機能を組み合わせる条件や処理を記述することができない．

6.2 携帯端末上のタッチパネルの操作に関する研究

携帯端末のタッチパネル上での目標に大きさに関する研究が行われている [15][17][16]．実験によりタッチパネル上では，目標が大きくなる程，選択の速度や向上することが分かった．

この中で Parhi らは，1 つの目標を選択する場合は 9.2mm 以上，連続して目標を選択する場合は 9.6 ｍｍ以上に目標の大きさを設定することを勧めている．本研究では，これらの実験を参考にして，コンポーネントの大きさを 10.0mm に決定した．

タッチパネル上でターゲット選択の際に，指で目標が隠されてしまう問題に対応する研究が行われている．Offset Cursor[18] や Shift[21] では，指によって隠される領域を指から離れた位置に表示することでこの問題に取り組んでいる．Escape[24] は，目標の形状が示す方向のジェスチャを行うことによって，目標と離れた位置からの選択を可能にしている．本研究では，これらの研究を参考とし，コンポーネントをドラッグアンドドロップする際に，指の位置から少しずらした位置にコンポーネントを表示するようにした．この方法は，Offset Cursor と同様であり，Offset Cursor には画面の端にある選択対象を選択できないという問題があるが，MoCoPro では，携帯端末の画面の端から離した位置に選択対象であるコンポーネントを自動配置するため問題とならない．

第7章　議論

7.1　プログラムの記述に伴う手間について

MoCoPro では，プログラムを記述するために，ユーザが１つ１つ機能を組み合わせ，データフロー図を記述する必要がある．このため，プログラムを記述することに時間がかかってしまうという問題がある．ユーザがコンテキストアウェアシステムを開発する方法としては，6.1 節においても述べたように，ビジュアルプログラミング以外にもメニュー方式や PbD などの手法がある．特に PbD は，プログラムを記述する必要がないため，ユーザへの負担が小さい．しかし，コンピュータが行う推論によりプログラムを記述するため，ユーザの要求とは異なる動作を行う可能性がある．このような場合に，コンピュータが出力したプログラムを要求に合致するように編集を行うことや，新たな条件や処理を追加できる環境があれば便利である．本研究においては，携帯端末上でコンテキスト依存プログラムを記述する環境を構築することに取り組んできたが，PbD と合わせて使用することや，現在や過去のコンテキストや操作の履歴を用いることで，プログラムを記述する手間を軽減し，より便利な環境に変更していくことができると考えられる．

7.2　プログラムの記述について

MoCoPro は，携帯端末を持っているような一般的なユーザである十代から五十代程度までのユーザがプログラムを記述できるようにすることを目指している．しかし，このようなユーザが MoCoPro を利用してプログラムを記述できるかどうかについては実験を行っていない．MoCoPro の ECA ルールに基づいて，コンテキスト定義とアクション定義に分割して記述する方式は，エンドユーザであっても理解できると考えている．これは，何らかの条件を満たした際に，動作を行うことは,「10 時になったら，電話をする」のように日常生活にありふれたものであるためである．データフロー図については，データの流れを示す簡易なモデルであるため，エンドユーザであっても理解が可能であると考えられる．現在の実装では，データフロー図の一部を実行できるようにし，確認を行える機能を設けているのみであるのが，処理が行われる過程を視覚的に表現することで，より分かりやすくすることができると考えられる．また，コンテキスト定義の AND と OR の関係の記述については，被験者実験を行い，エンドユーザにとってより分かりやすい表示を行う必要があると考えられる．

7.3　画面設計と操作インタフェースについて

MoCoPro では，一画面に複数のデータフロー図を作成することによりコンテキスト定義やアクション定義を記述する．プログラムの大きさは，5.2 章で示したプログラム程度を想定しており，1 つの記述の大きさは，5 個程度のコンポーネントで構成されることが多いと考えられる．現在の環境では，一画面で表示できるコンポーネントの数は，携帯端末の画面の長辺方向には 5 つ，短辺方向には 4 つ程度である．5 個程度のコンポーネントの場合は，縦横に 2 ないし 3 コンポーネント程度となるため，1 つの記述全体を一画面に見ることは問題なく行えると考えられる．また，タッチパネルを使用しているため，スクロール操作が行いやすく画面の移動が負担にならないことに加え，フォルディング操作により，プログラムを小さく表示することもできるため，他の記述との関連も確認しやすく，画面の大きさの問題にならないと考えられる．

画面の切り替えに用いるクロッシング操作は，画面を小さくすることなく，操作も行いやすいように設計されている．しかし，クロッシング操作は，一般的に使用される操作ではないため，現在の影でクロッシング操作が行えることを示す表示をより分かりやすくする必要はあると考えられる．

また，携帯端末は機種によって，画面の大きさや形状が様々であるため，機種ごとの調節が必要になる場合が多い．MoCoPro では，タッチパネルを搭載していることが必須となることに加え，iPhone や iPod touch 以外の環境に移行した際にいくつか変更を行う必要があると考えられる．まず，画面の大きさが小さくなった場合を考える．Parhi らの実験 [15] によると，画面の要素は，9.6mm 以上が望まれるため，画面が小さくなった場合は，画面の構成要素を大きくする必要がある．現在の実装では，コンポーネントの大きさを 10.0mm としており，また，プログラムを記述する画面にも若干の余裕があるが，画面が小さくなると操作が行いにくくなることは避けられない．しかし，タッチパネルを搭載した携帯端末の普及に伴い，大きな画面を持った携帯端末が増加しているため，今後問題は解消されていくと考えられる．また，クロッシング操作も影響を受ける可能性がある．クロッシング操作は，携帯端末の画面の端を横切る動作を行う必要があるが，画面の淵に段差がある携帯端末が多くあり，この場合は，クロッシング操作を行えないという問題がある．この問題に対応するためには，画面の淵の段差に沿って指を滑らすことでことで同様の操作を行うなどの変更が必要になる．

7.4　プログラミング環境の実装について

MoCoPro は，新たなコンポーネントを追加することができるが，様々な機能を実現するために，コンポーネントの数が増えてしまうという問題が起こる可能性がある．ユーザがコンポーネントを選択する際のことを考えると，コンポーネントセレクタは階層構造を持つので，階層を増やし分類を行えば，目的のコンポーネントを探しやすくすることができると考えられる．

MoCoPro のコンテキストの条件の記述には，様々な機能を組み合わせることが可能である．例えば，場所のコンテキストの条件に，集合場所を記したメモを読み込んで得られる場

所を指定することが可能である．しかし，メモを読み込む処理を GPS センサから信号を受信するたびに行うのは効率が悪いため，現在の実装では一定間隔（５分）ごとにコンテキストの条件を更新するためのイベントを発生させている．一定間隔ごとにすべての更新を行うことについても効率が悪いと考えられるので，ユーザが更新の間隔を指定できるようすることや，使用している機能自体が更新の必要をプッシュにより知らせられるようにするなどの変更が必要であると考えられる．

第8章　おわりに

本研究では，コンテキストアウェアシステムに対するユーザの多種多様な欲求に対応するため，ユーザが自分自身でコンテキストを利用するプログラムを作成することを可能にすることを目的として，携帯端末上でコンテキスト依存プログラムを記述するための環境であるMoCoProを提案し，実装した．MoCoProは，ビジュアルプログラミング形式を採用し，プログラムを視覚的に表現することによって，携帯端末を使用する幅広いユーザがプログラムを記述できるようにした．また，ECAルールを参考にし，プログラムの記述をコンテキスト定義とアクション定義に分割して行うことで，エンドユーザであっても理解しやすくし，また管理を行い記述方式を示した．携帯端末の入力インタフェースとしてタッチパネルを採用し，タッチパネル上でビジュアルプログラミングを行うために適したインタフェースや操作方法を示した．

謝辞

本研究を進めるにあたり，御指導を頂きました田中二郎教授に，心より深く感謝致します．また，志築文太郎講師には，研究活動の全般において，本当にきめ細やかな御指導を頂きました．心より感謝致します．さらに，三末和男准教授，高橋伸講師には，論文の執筆において様々な助言を頂きました．深く感謝致します．WAVE チームの皆様をはじめ，IPLAB の皆様には，研究のみならず，私生活においても良い刺激を与えてくれました．ありがとうございました．そして，いつも私を力強く支えてくれる家族や友人達にも感謝を申し上げたいと思います．本当にありがとうございました．

参考文献

[1] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, Vol. 3, No. 5, pp. 421–433, 1997.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pp. 304–307, London, UK, 1999. Springer-Verlag.

[3] Johnny Accot and Shumin Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 73–80, New York, NY, USA, 2002. ACM.

[4] Sheetal Agarwal, Anupam Joshi, Tim Finin, Yelena Yesha, and Tim Ganous. A pervasive computing system for the operating room of the future. *Mobile Networks and Applications*, Vol. 12, No. 2-3, pp. 215–228, 2007.

[5] Shea Armstrong, Yael Kollet, and Trevor J. Smedley. Visual scripting for handheld computers. *Human-Centric Computing Languages and Environments, IEEE CS International Symposium on*, p. 68, 2002.

[6] Parks Associates. GPS: A path to new applications on mobile devices. *http://www.parksassociates.com/*.

[7] David Bannach, Oliver Amft, and Paul Lukowicz. Rapid prototyping of activity recognition applications. *IEEE PERVASIVE COMPUTING*, Vol. 7, No. 2, pp. 22–31, 2008.

[8] Philip T. Cox, F. R. Giles, and T. Pietrzykowski. Prograph: A step towards liberating programming from textual conditioning. In *Visual Languages IEEE Workshop*, pp. 150–156, Los Alamitos, California, 1989. IEEE CS Press.

[9] Umeshwar Dayal. *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[10] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a CAPpella: programming by demonstration of context-aware applications. In *CHI '04: Proceedings*

of the SIGCHI conference on Human factors in computing systems, pp. 33–40, New York, NY, USA, 2004. ACM.

[11] E. J. Golin and S. P. Reiss. The specification of visual language syntax. In *Journal of Visual Languages and Computing*, Vol. 1, pp. 141—157, 1990.

[12] Yael Kollet and Trevor J. Smedley. Message-flow programming in pdagraph. *Visual Languages - Human Centric Computing*, pp. 229–232, 2004.

[13] Panu Korpipaa, Esko-Juhani Malm, Ilkka Salminen, Tapani Rantakokko, Vesa Kyllonen, and Ilkka Kansala. Context management for end user development of context-aware applications. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pp. 304–308, New York, NY, USA, 2005. ACM.

[14] Brad A. Myers. Taxonomies of visual programming and programming visualization. In *Journal of Visual Languages and Computing*, Vol. 1, p. 1, 1990.

[15] Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pp. 203–210, New York, NY, USA, 2006. ACM.

[16] Yong S. Park, Sung H. Han, Jaehyun Park, and Youngseok Cho. Touch key design for target selection on a mobile phone. In *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pp. 423–426, New York, NY, USA, 2008. ACM.

[17] Keith B. Perry and Juan Pablo Hourcade. Evaluating one handed thumb tapping on mobile touchscreen devices. In *GI '08: Proceedings of graphics interface 2008*, pp. 57–64, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[18] Richard L. Potter, Linda J. Weldon, and B. Shneiderman. Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 27–32, New York, NY, USA, 1988. ACM.

[19] Albrecht Schmidt, Michael Beigl, and Hans w. Gellersen. There is more to context than location. *Computers and Graphics*, Vol. 23, pp. 893–901, 1999.

[20] International Telecommunication Union. Worldwide mobile cellular subscribers to reach 4 billion mark late 2008.

[21] Daniel Vogel and Patrick Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 657–666, New York, NY, USA, 2007. ACM.

[22] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, Vol. 10, No. 1, pp. 91–102, 1992.

[23] Torben Weis, Mirko Knoll, Andreas Ulbrich, Gero Mhl, and Alexander Brndle. Rapid prototyping for pervasive applications. *IEEE PERVASIVE COMPUTING*, Vol. 6, No. 2, pp. 76–84, 2007.

[24] Koji Yatani, Kurt Partridge, Marshall Bern, and Mark W. Newman. Escape: a target selection technique using visually-cued gestures. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 285–294, New York, NY, USA, 2008. ACM.

[25] Youngwoo Yoon, Yuri Ahn, Geehyuk Lee, Sungmoo Hong, and Minjeong Kim. Context-aware photo selection for promoting photo consumption on a mobile phone. In *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pp. 33–42, New York, NY, USA, 2008. ACM.

[26] 総務省情報通信政策局. 通信利用動向調査報告書 （世帯編）平成 19 年度調査. *http://www.soumu.go.jp/*.

[27] 寺田努, 宮前雅一. その場プログラミング環境の実現に向けて. 情報処理学会研究報告. UBI, [ユビキタスコンピューティングシステム], Vol. 2007, No. 46, pp. 1–8, 20070523.

[28] 加藤文彦, 遠山緑生, 服部隆志, 萩野達也. 携帯端末上での曖昧なコンテクストに基づく例示プログラミング. 第 7 回プログラミングおよび応用のシステムに関するワークショップ (SPA2004).

[29] 西本裕貴, 志築文太郎, 田中二郎. 携帯端末上でコンテキスト依存プログラムを記述するためのビジュアルプログラミング環境. 第 16 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2008), pp. 145–146, 2008.

[30] 西本裕貴, 志築文太郎, 田中二郎. 携帯電話上でコンテキスト依存プログラムを記述するためのビジュアルプログラミング環境. 情報処理学会第 70 回大会, 2008.