修士論文　　2002年度　　(平成14年度)

情報端末 SmartTerminal 構築のための基盤機構の実装

慶應義塾大学大学院政策・メディア研究科
石井 かおり

修士論文要旨　　2002年度　　(平成14年度)

情報端末 SmartTerminal 構築のための基盤機構の実装

論文要旨

近年、計算機の小型化やネットワーク技術が発展し、ユーザは移動中に計算機やインターネットを利用するようになった。このため、駅やファーストフード店など公共な空間では無線通信を用いてインターネットに接続するホットスポットサービスが開始された。ユーザはノートパソコンなど携帯端末を持ち歩くことで、インターネットを利用できる。しかし、ホットスポットサービスを利用するにはユーザはノートパソコンを持ち歩かなければならず、利用可能な場所にもまだ限りがある。

本研究では、ユーザの屋外での計算機利用活動を支援するために、情報端末 SmartTerminal を提案する。 SmartTerminal では通信、情報、機器機能を提供し、ユーザが多くの機器を持ち歩かなくても環境で提供可能する。SmartTerminal では、多数のサービスが提供されることが想定されるため、サービスが容易に組み替えられる必要がある。そのため、SmartTerminal 上のサービスを提供、管理するための基盤機構 SmartTerminal Framework を構築した。SmartTerminal Framework ではサービス、クライアント、通信と位置アプリケーションを実装するためのインタフェースを提供する。それらを実装したアプリケーションは、SmartTerminal Framework 基盤機構で管理でき、透過的に扱うことが可能になる。

本論文では、SmartTerminal を構築する上で必要となる基盤技術を述べ、本研究で提案する 情報端末 SmartTerminal のモデルを述べる。また、設計および実装を述べ、アプリケーション例として SmartTaxi アプリケーションをあげる。評価および関連研究をあげて、本論文をまとめる。

キーワード

1 ユビキタスコンピューティング 2 ホットスポットサービス 3 公共情報端末 4 携帯端末 5 位置情報

慶應義塾大学大学院政策メディア・メディア研究科

石井 かおり

Abstract of Master's Thesis

Academic Year 2002

Design and Implementation of Middleware for SmartTerminals

Summary

Developments in computer and network technology has enabled downsizing of portable devices, making it easier for users to use computers and the Internet on the go. In hotspots located in stations and fast food restaurants, users may connect their laptop computers to the Internet by using the wireless LAN provided there. Unfortunately, in order to receive these services, users are burdened by the need to carry their laptops.

This research aims to provide users with computer resources in public space by the use of SmartTerminals. A SmartTerminal is a terminal equipped with communication, information and device functions to assist users so they do not always need to carry heavy laptops nor equiptment. Services installed on SmartTerminals are provided by various vendors, making it necessary for a framework to enable easy installation of services. This is realized by using SmartTerminal Framework, which defines service interface, and manages all resources on the SmartTerminal such as devices, software, and location by making them modular and recomposable.

This paper presents the basic technology and charactoristics of a SmartTerminal and our approach for design and implementation. As an application built on top of the framework, we give SmartTaxi.

Key Words

1 Ubiquitous Computing 2 Hotspot 3 Smart Terminal 4 Portable Device 5 Location

Keio University Graduate School of Media and Governance

Kaori Ishii

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

近年，小型携帯端末の普及に伴い，人々の計算機の利用環境が屋内から屋外へと広がっている．従来の計算機は大きく，ネットワークへの接続方法が有線のみであったため，持ち歩くことが困難であった．計算機が小型化し，無線技術が普及したことで，計算機を移動中にネットワークに接続した状態で利用することが可能になった．

中でも，携帯電話はウェブページの閲覧が可能など利便性が高いため数年の間に普及し，図 1.1 のグラフに示すように一般家庭におけるパソコンの保有率を上回っている [1]．携帯電話を利用することで，人々は屋外でもインターネットに接続してオンラインバンキングなど様々なサービスを受けられる．携帯電話は単にコミュニケーションを行う道具としてだけではなく，周辺の機器との協調や現在位置の取得のように幅広い利用が可能になっている．

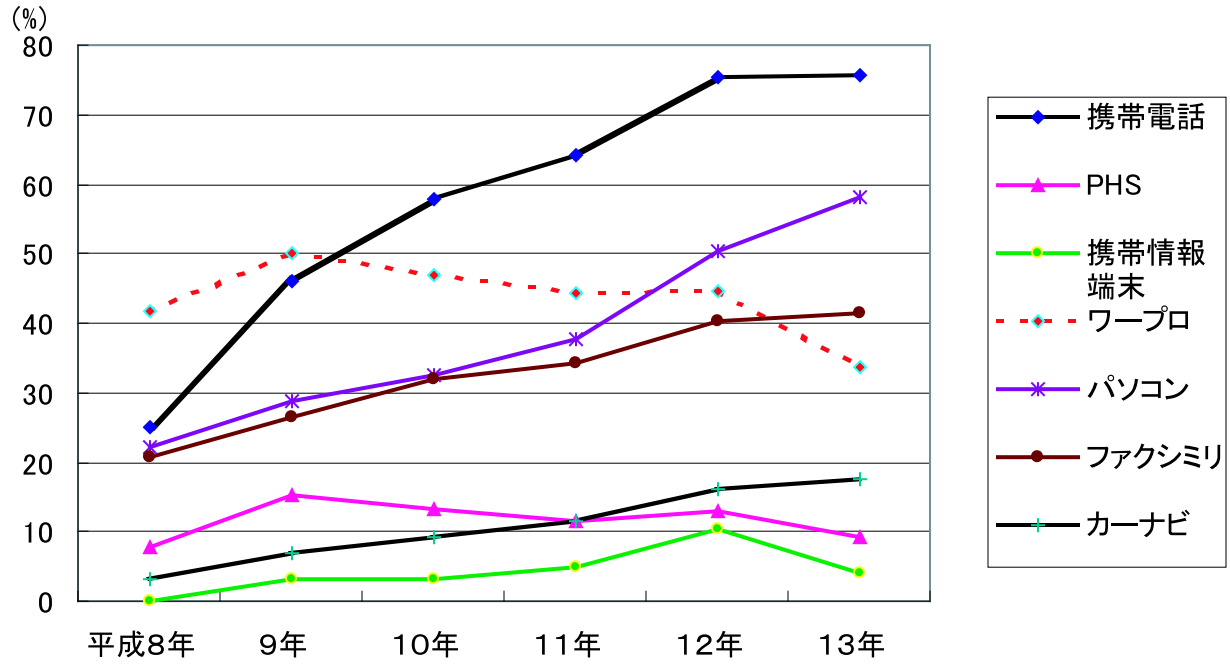

図 1.1: 情報通信機器の世帯保有率

また，無線機器が安価で手に入るようになり，無線ネットワークおよびそれを用いて計算機をインターネットに接続するユーザが増加している．この動向に注目し，複数の企業は人が多く集まる駅やファーストフード店にて**ホットスポットサービス**の利用実験およびサービス提供を開始している [2][3][4]．ホットスポットサービスとは，インターネット接続を提供するサービスである．利用者はノートパソコンをホットスポットサービスに持ち込むことで，インターネットサービスを利用できる．

このように，人々の計算機資源およびそれを支える利用環境やインフラが徐々に豊かになっている．計算機は，パソコンや携帯電話以外にも情報家電機器など多様な機器に埋め込まれている．さらにこれらの計算機にはネットワーク接続が可能なものもあり，ネットワークインフラを利用して相互に接続する．今後，人々に計算機とネットワークインフラを組み合わせた様々なサービスが提供可能になる．現在は，携帯電話と無線環境の利用が注目を浴びているが，これらには以下のような 3 点の問題点がある．

1.1.1 携帯電話の入出力機能の制約

携帯電話を利用することでユーザは移動しながら通信を行い，インターネットを利用できるようになった．利用可能なコンテンツも静止画から動画へ広がり，提供されるサービスも充実した．しかし，端末が小型であるためユーザは多くの制約を強いられる．インターネット機能のある携帯電話を所有するインターネット・ユーザを対象に行ったアンケート [5] の，携帯電話での情報収集・サービスを利用したくないと回答したユーザの理由を図 1.2 に示す．このように，画面が見辛い，入力などの操作性が悪いなど主要な入出力の問題点があげられている．サービスが充実してユーザが扱うデータ量が増加するほど入出力の制約は問題となる．

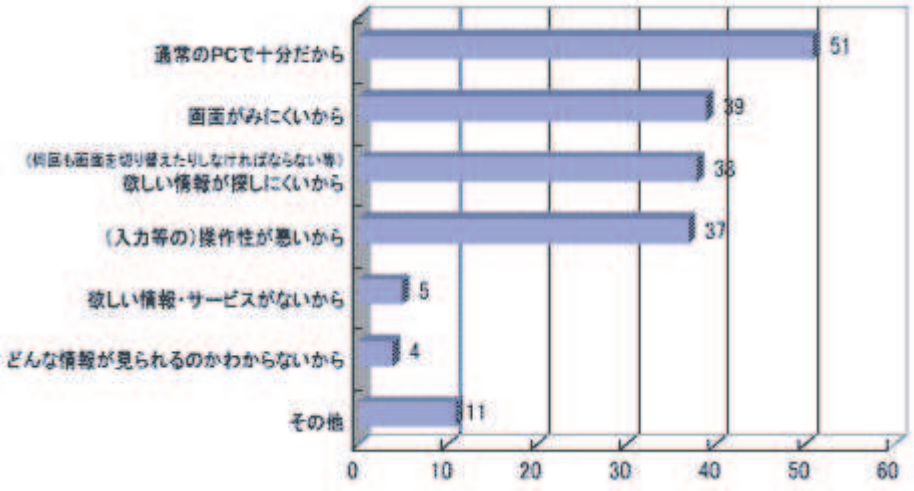

図 1.2: 携帯電話端末からの情報収集を利用しようと思わない理由

1.1.2　機器持ち運びの煩雑性

ホットスポットサービスを利用するには，ユーザはノートパソコンを持ち歩く必要がある．ノートパソコンが軽量化して計算機環境を持ち歩くことは容易になったが，実際に移動中にノートパソコンを利用するのは煩わしいと感じるユーザも少なくない．ノートパソコンを携帯して外出先からインターネットに接続しているユーザを対象に行ったアンケート [6] の，不満に感じてるあるいは問題だと感じてることに対する調査結果を図 1.3 に示す．このように，ホットスポットサービスの利用に関しては周辺機器の持ち運びは困難，インターネット接続料金が高い，電源の確保が難しいなどの問題がある．

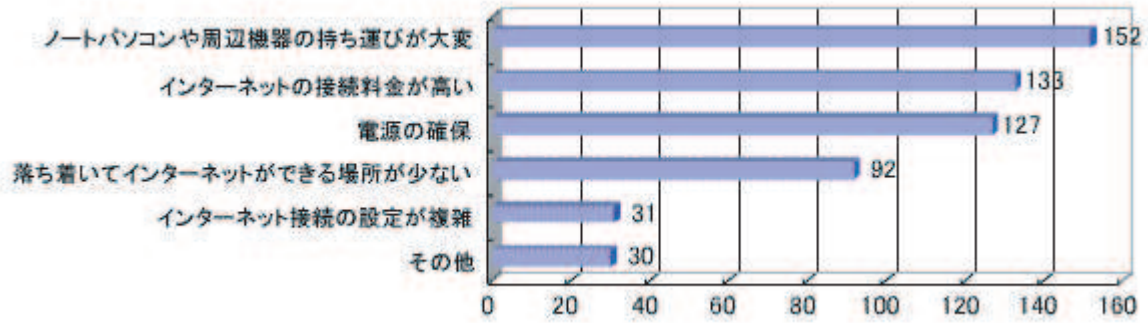

図 1.3: ホットスポット利用の弊害

1.1.3　サービスの解離

従来，サービスを受けるためにはユーザはそのサービスと物理的に空間を共有する必要があった．サービスや情報のオンライン化により，実際には空間を共有しなくとも遠隔から情報をアクセスしてサービスを受けることが可能になった．ユーザはパソコンや携帯電話からインターネットのアドレスを指定するため，場所に関係なくサービスを受けられる．しかし，サービスの端末を目の前にして直接利用できる距離にいるユーザさえも，わざわざネットワーク上の情報をアクセスする必要がある．この間接的な利用方法を図 1.4 に示す．全てのサービスをオンライン化してしまうと，物理空間で近いものが実際には遠くなってしまう問題がある．また，現在の携帯電話の利用モデルでは，情報をアクセスするたびに課金されてしまう．

1.2　本研究の目的と概要

本研究では，携帯端末と無線環境の問題点に着目し，公共な空間において通信，情報および機器サービスを提供可能な情報端末 **SmartTerminal** のモデルを提案し，その構築に必要となる基盤機構の設計および実装を行う．

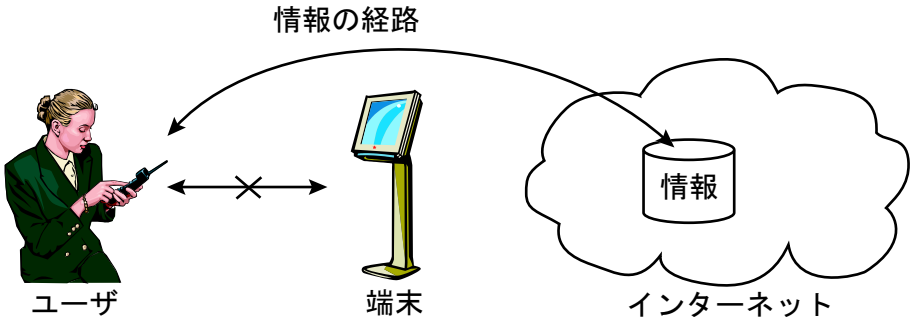

図 1.4: 遠隔にある情報の利用

ユーザが携帯電話を用いてマルチメディアサービスを利用する場合でも，携帯電話の限られた入出力機能の使用を強いられる．また，現状のホットスポットサービスでは，インターネット接続のみ提供しているためユーザはノートパソコンを持ち歩く必要がある．機器に関する問題以外に，ユーザは目の前のサービスを利用するためにインターネットを用いて遠隔のサービスをアクセスする必要がある．本研究で提案する SmartTerminal では，入出力が可能な機器サービスを提供することで機器の制約や持ち運びの煩雑性を軽減する．また，図 1.5 に示すように，機器端末を SmartTerminal 化して通信機能を備えることで，ユーザが直接遠隔の情報をアクセスせずにサービスを利用できる．このように SmartTerminal ではユーザが公共の環境で計算機利用活動を行うために必要な通信，情報，機器機能を提供する．

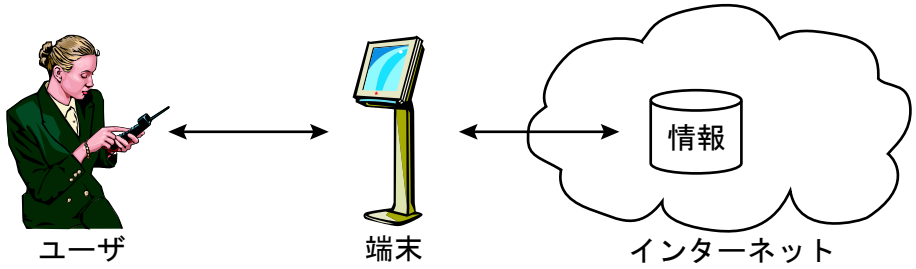

図 1.5: サービスの間接利用

また，SmartTerminal を構築するためには，多種多様なサービスを実装し管理するための基盤機構が必要となる．SmartTerminal は公共な空間に設置されるため，提供するサービスは提供者，地域，端末によって異なることが想定される．しかし，サービスごとに操作方法，管理方法などのインタフェースが異なると，SmartTerminal で各サービスのインタフェースの差を隠蔽する必要があり，導入コストが高くなってしまう．そのため，SmartTerminal に携帯端末およびネットワークとの通信，提供サービス，クライアント認証などのサービスのインタフェースを統一し，保守運用が容易にできる基盤機構が必要となる．本研究では SmartTerminal を構築するための基盤機構 SmartTerminal Framework を提供する．サービス提供者が SmartTerminal Framework を用いてサービスを実装する

ことで，サービスが透過的に扱え，SmartTerminal が容易に構築できる．

1.3 本論文の構成

本論文では，提案する情報端末 SmartTerminal のモデルを提案し，基盤機構となる SmartTerminal Framework および応用アプリケーションについて述べる．第 2 章では現在の公共の環境やそこに存在するさまざまな要素と問題点を述べる．第 3 章では情報端末の現状と SmartTerminal の機能要件および先行技術について述べる．第 4 章，第 5 章では本 SmartTerminal Framework の設計および実装を示し，第 6 章では SmartTerminal Framework を用いた応用アプリケーション例を示す．第 7 章，第 8 章では本機構の評価および関連研究を挙げ，第 9 章で今後の課題を整理する．

第2章　公共空間の計算機環境とサービス

本章では，公共空間およびそこに存在するサービスの視点から特性を検証する．まず，公共空間における計算機利用環境の特徴を挙げ，環境の基盤となる通信手段を説明する．次に，それらの環境に存在するサービスの特徴を検討し，更に公共空間におけるホットスポットサービスの必要性を述べる．

2.1　公共空間の計算機環境とサービス

人々が行き交い生活する場所の 1 つとして街などの公共な空間がある．そこには公共機関，公共施設，交通機関や店など人々の生活を支援するサービスが多数存在する．近年これらのサービスに計算機やネットワークが導入され，サービスの運用形態や利用形態が変化した．例えば，従来の行政手続をオンライン化した電子政府・電子自治体が施行段階にある [7]．行政手続きをオンライン化することで，申請や届出の手続きを自宅又は会社からインターネットでてき人々の利便性が高まり，作業の効率化も可能になる．また，サービスのオンライン化だけでなく，計算機とインターネットを利用する環境が屋外の公共な空間へも広がった．人々はノートパソコン，PDA，携帯電話といった移動性の高い機器を多く持ち歩き，これらを移動しながら利用して継続的にサービスを利用できる．

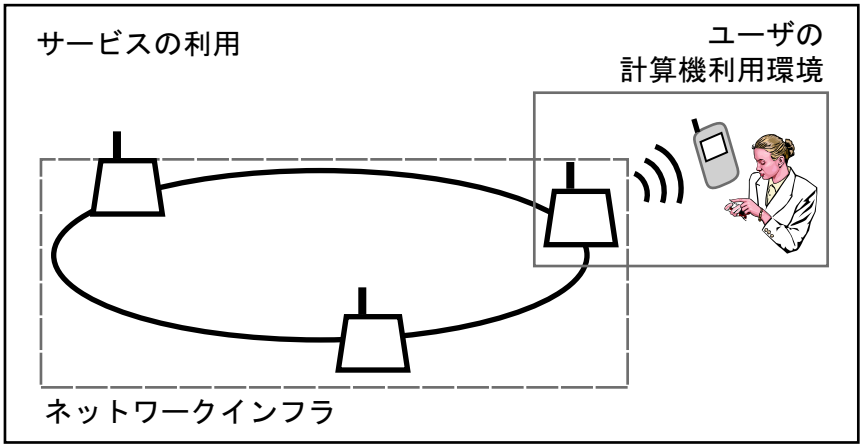

図 2.1: サービスの利用形態

このように，計算機とネットワークが多様なものに取り込まれ，ユーザの計算機利用形態や取り巻く環境が変化した．現在の公共な空間におけるサービスの利用形態は，図 2.1 で示す通り，サービス，ネットワークインフラとユーザの計算機利用環境の 3 つに分けられ，以下にそれらを示す．

計算機利用環境 周辺に機器や計算機が多数存在し，ユーザがそれらを利用する活動環境を示す．また，サービスを提供する機器をサービス端末，ユーザが保持する携帯電話などをクライアント端末と呼ぶ．

ネットワークインフラ 計算機環境を構築するために必要なインターネットのネットワーク基盤を指す．

サービス ユーザに物，情報や手段を提供する公共サービスや交通機関などを指す．

次の節以降で，それぞれの側面について要素技術や例をあげて説明する．

2.2 計算機利用環境

従来，ユーザのコンピューティング環境はオフィスや大学など屋内がほとんどであった．その主な原因としては計算機そのものの物理的な大きさや，ネットワークに有線で接続されているなどがあげられる．しかし，計算機が小型化し，ノート型パソコンや PDA などの小型携帯端末が登場したため，ユーザはこれらを持ち運び移動先で利用でき，移動性が高まった．更に無線技術が発展したため，有線に加えて無線を用いてネットワークが構築可能になった．計算機の通信方法が有線から無線に移行し，ユーザの移動性は更に高まった．この移動性に着目し，複数の企業が公共空間で無線によるネットワーク接続サービスを開始した．JR 東日本では無線による駅でのインターネット接続実験 [8] を行い，モスバーガー店舗内には NTT コミュニケーションズ (株) が提供する高速無線 LAN 接続サービス「ホットスポット」[2] がサービスを提供している．

計算機の移動性が高まり，計算機の利用環境は閉じられた空間から公共空間へと広がった．従来の計算機環境では，空間，サービスとユーザをある程度限定することが可能であった．計算機の利用環境が公共な空間に広がったことにより，サービスとユーザが動的に変化するため，それぞれを特定することが困難になる．そのため，動的に変化するサービスとユーザを特定し，継続的に通信を行うための考慮が必要となる．計算機端末の移動性とネットワーク接続性を想定したコンピューティング環境にはユビキタスコンピューティング環境とモバイルコンピューティング環境の 2 つがある．それぞれを図 2.2 に示し，以下で説明する．

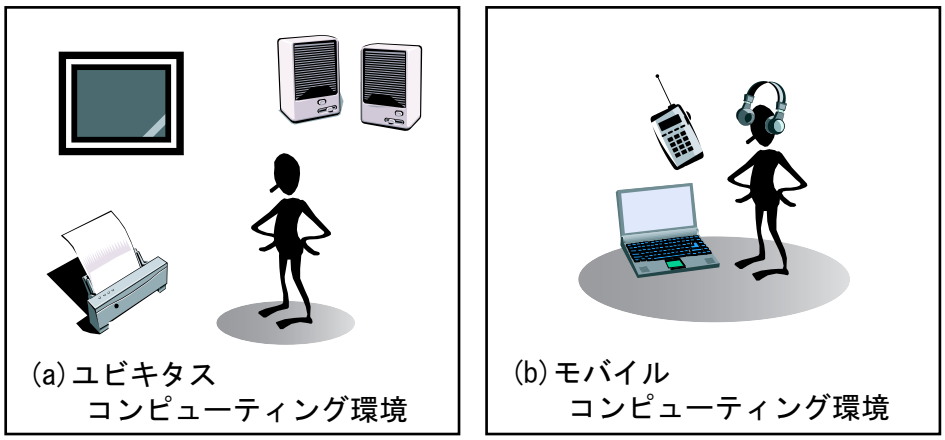

図 2.2: コンピューティング環境

ユビキタスコンピューティング環境

計算機の小型化が進み，車や家電製品などさまざまな物に計算機が組み込まれるようになった．計算能力だけではなく，ネットワーク接続機能を持つもの

もあり，これらをネットワークに接続してネットワークを介して利用できる．計算機だけでなく，センサも小型化し，多様なセンサが環境に埋め込まれ，環境の状態を取得し，その情報を提供できる．このような計算機やセンサが至るところに遍在しどこからでもアクセス可能な環境を**ユビキタスコンピューティング環境**と呼ぶ．

ユビキタスコンピューティング環境の 1 例として，情報家電機器ネットワークがある．情報家電機器ネットワークでは，テレビ，電子レンジや冷蔵庫などの機器および家庭内の状態を取得するためのセンサ群がネットワークに接続され，家庭内のみならず外出先から制御可能である．情報家電機器ネットワークの実現例として，電子情報技術産業協会の情報家電モデルハウス [9] や，ユビキタスコンピューティング環境を街全体に応用した Hewlett-Packard 社の cooltown[10] プロジェクトなどがある．

モバイルコンピューティング環境

計算機や周辺機器間の連携を考慮することでユーザの移動透過性を実現したコンピューティング環境を**モバイルコンピューティング環境**と呼ぶ．モバイルコンピューティングでは，ユーザの移動中のコンピューティングを支援するために 2 通りの方法をとる．1 つ目は，あらゆる機器を持ち運び可能にしてユーザがそれらを持ち歩き，2 つ目は，ユーザは最低限必要な機器のみを持ち歩く．前者の例として，ユーザが移動中にメールを読み，スケジュールを確認し，書類を作成し，音楽を聞くといった複数の作業をしたいと想定する．この場合，必要となるノートパソコン，携帯電話，ヘッドフォンなど全ての機器をユーザが常に持ち歩く．後者では，ユーザは携帯電話や PDA など必要最低限の機器のみを持ち歩き，大きなディスプレイや音響環境が必要となれば，それらは周辺の環境から補う．

モバイルコンピューティング環境の例として，携帯電話やホットスポットサービスが挙げられる．インターネット対応型携帯電話を利用することでユーザは移動中もインターネットが利用でき，作業の継続性が保たれる．また，ユーザは無線ネットワークカードを持つノート型パソコンをホットスポットサービスに持ち込めば，そこで提供されているサービスを利用できる．

これらの計算機利用環境が実現されれば，ユーザは移動しながら継続的にサービスを受けられ，周辺に存在するさまざまな機器や計算機を利用できる．しかし，実際にネットワーク接続機能を持つ機器は多く存在せず，周辺の環境でユーザに提供されている機器や計算機資源も数少ない．ユーザの計算機利用環境が公共の空間に広がるとともに，公共な空間で提供するユーザが利用可能な機器やサービスを増やす必要がある．

2.3 ネットワークインフラ

ユーザの公共空間における計算機利用環境を支援する要素としてネットワークインフラがある．ユーザがサービスの情報を取得するためには，情報を提供しているインターネット上の計算機と通信する必要がある．計算機や端末間を相互に接続しネットワークを構築することで，これらの環境において通信を確立できる．ネットワークの形態としては大まかに有線ネットワークと無線ネットワークがあげられる．有線ネットワークは光ファイバーやイーサネットなどの実線が形成するネットワークを指し，主に日本国内を結ぶインターネットのバックボーンに用いられている．無線ネットワークは，実線を使わずに電波を利用して計算機間で通信を行い，移動や再構成が頻繁に起るネットワークを構築する手段として用いられる．

公共空間の計算機利用環境を構築するためには，サービスの情報を管理するサーバとユーザに情報を提供する端末間を接続する必要があり，これには有線ネットワークが利用される．また，サービス端末とクライアント端末間は，クライアント端末は移動型であるため，無線ネットワークが利用される．このように，サービス端末は 2 つのネットワーク形態を持つ必要がある．無線ネットワークを構築するためには，無線通信媒体とインターネットを利用するサービスを考える必要がある．第 2.3.1 項と第 2.3.2 項でそれぞれを説明し，第 2.3.3 項でそれらを利用したサービスとしてホットスポットサービスをあげる．

2.3.1 無線通信媒体

サービス端末とクライアント端末間の通信を行う方法として無線通信が用いられる．無線によるネットワークアクセス手段には複数あり，表 2.1 に示す．それぞれ周波数によって用途が割り当てられ，最大通信速度が異なる．

表 2.1: 無線によるネットワークアクセス

周波数帯	用途	最大通信速度
800MHz	携帯電話	9.6Kbps
1.9GHz	PHS	64Kbps
2GHz	IMT-2000	2Mbps(静止時)
2.4GHz	無線 LAN(IEEE802.11b)	11Mbps
2.4GHz	Bluetooth	1Mbps
5.2-5.8GHz	無線 LAN(IEEE802.11a)	6/12/24Mbps

これらのうち，現在近距離通信を行う手段として利用されてる媒体をあげる．

IEEE802.11

IEEE[11] では無線 LAN の標準化を進めており，IEEE802.11[12] 規格が主流となっている．IEEE802.11 では，無線 LAN のパケット通信アクセス方式として搬送波感知多重アクセス/衝突回避を利用する．IEEE802.11 規格の中でも，IEEE802.11a や IEEE802.11b があり，上の図で示したとおり，前者は 5.2-5.8GHz の周波数帯を利用し，通信速度は 6/12/24Mbps である．後者は，2.4GHz の周波数帯を利用し，通信速度は 11Mbps である．

赤外線通信

赤外線通信は赤外線を利用して無線通信を行う手段であり，赤外データ通信の規格として IrDA[13] がある．赤外線は直進性が高く，通信を行うためには機器同士が一直線上にあることが望ましい上，天候や障害物に弱い．物によっては数 km 離れた場所と通信もできるが，完全な見通しが必要である．

Bluetooth

Bluetooth[14] も赤外線同様近距離通信を行う手段であり，複数の企業が先導して Bluetooth Special Interest Group を構成している．Bluetooth は IEEE802.11b と同様，2.4GHz の周波数帯を利用し，10 メートル以内の距離では 1Mbps の通信速度を達成する．現在，携帯電話やノート型パソコンなどに搭載されて即興的な端末間の通信に利用されている．

2.3.2 インターネットサービスプロバイダ

インターネットサービスプロバイダ (ISP) は，家庭や企業とインターネットのバックボーンを接続し，インターネット接続サービスを提供する．インターネットを利用するためには ISP と契約する必要がある．現在，加入電話会社が提供するサービス，携帯電話や PHS から利用可能なサービスなど多くの ISP が存在する．しかし，ISP は加入電話を単位にサービスを契約するため，携帯電話を利用しなければ移動中は容易に ISP に接続することができない．

2.3.3 ホットスポットサービス

第 2.3.1 項と第 2.3.2 項であげた無線通信媒体と ISP を組み合わせたサービスとしてホットスポットサービスがある．現在，駅，ファーストフード店やホテルなど公共な空間がホットスポットサービス化されている．サービス提供者は ISP と契約し，店内に無線基地

局を設置する．その際ホットスポットサービスとISPは，図2.3に示すように1対1または1対多の関係にある．1対1のサービスでは，ユーザは利用するためにそのサービスが提携しているISPと契約しなければならないものがある．1対多のサービスは，国際標準ローミングによりISPを限定せずに利用できるものを想定している．この場合，サービスを利用するためにユーザは新たにISPと契約する必要はない．ホットスポットサービスを利用する際に，ユーザは無線通信機能を備えたノートパソコンを持ち込む．

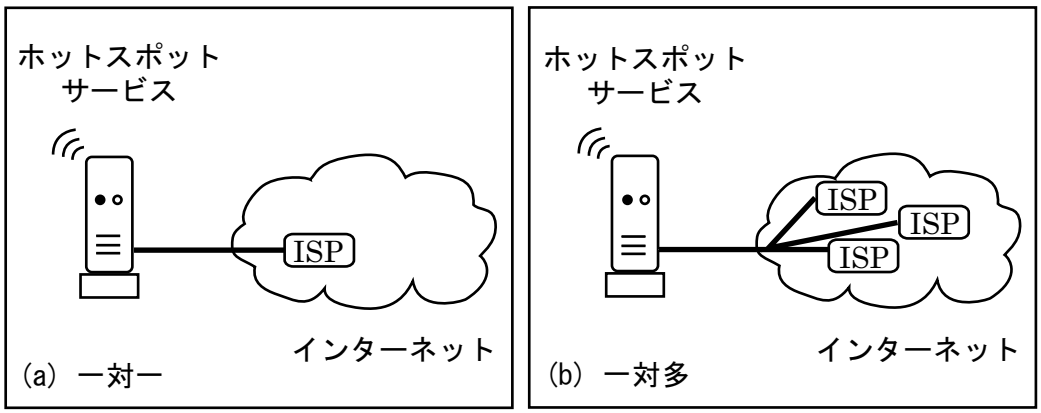

図 2.3: ホットスポットサービスとISP

2.3.4 ネットワークインフラを利用したサービス

公共な空間において，ユーザのネットワークを利用した計算機利用活動を実現するサービス端末を構築するには，インターネットに接続するためのISPと契約する必要がある．さらに，ユーザがクライアント端末からサービス端末を利用できるように，無線通信を可能にする必要がある．第2.3.3項で述べたホットスポットサービスも，公共な空間でネットワークに接続した計算機環境をユーザに提供する．しかし，これらを利用する場合には前述した機器の持ち運びの煩雑性の問題がある．サービス端末を構築する際はこれらの問題点を考慮する必要がある．

2.4 サービス

公共な空間では，多種多様なサービスが提供されている．それらが提供する物や形態はそれぞれ異なる．本節ではサービスの分類と利用形態を検討する．

2.4.1 サービスの種類

公共な空間で提供されているサービスは以下の表2.2に示すとおり4種類に分類できる．これらはユーザに目に見える物や目に見える形でサービスを提供する．

表 2.2: 公共空間で提供されるサービスの分類

サービスの種類	説明	例
交通サービス	交通機関など移動サービスを提供	バス，電車，タクシー
公共施設サービス	市や地方自治体が運営する機関	郵便局，図書館
商品サービス	服やジュースなど商品を提供	店，自動販売機
通信サービス	電気通信の手段を提供	公衆電話

また，これらのサービスに第 2.1 節で述べたように計算機やネットワークが導入され，サービスのオンライン化が進められている．これらのサービスは以下の表 2.3 のような構成要素に分けられる．

表 2.3: 提供されるサービスの構成要素

サービスの要素	説明	例
情報	情報を提供	オンラインサービス
機器	入出力の手段を提供	印刷，入出力装置

本研究では，表 2.2 で示したサービスのうち，オンラインでサービスを提供するものを対象とし，サービス提供を行うための要素として情報，機器および通信を考慮する．次に，サービスの例をあげる．

交通サービス 電車やバスなどの交通機関は，人々に移動する手段を提供する．これらのサービスはインターネットを利用して時刻表や運行状況などの情報を発信している．以前は時刻表をあらかじめもらっておく，あるいは停留所で確認する必要があった．情報のオンライン化により，コンピュータやインターネット対応の携帯電話端末を利用して容易にどこからでも時刻表を得られるようになった．

公共施設サービス 各市町村には，役所や図書館など様々な施設がある．電子政府の取り組みにも見られるように，公共のサービスにおいても情報や手続きのオンライン化が進められている．人々は，各自のパソコンからオンラインサービスを利用できる．また，必要に応じて印刷などの入出力機能を利用する．

商品サービス 商品を提供するサービスとして，食品店や衣服店があり，これらの中にはインターネットを利用してオンラインショッピングを可能にしている．これらのサービスは，カタログなどの情報を提供する．

通信サービス 街中で通信を提供するサービスには公衆電話がある．公衆電話自体は情報を提供しないが，電話機能以外にもパソコン通信機能などの通信機能を備えている．

2.4.2　サービスの形態

ユーザ，サービスとネットワークインフラを構成要素として，サービスの形態を図 2.4 で示す 4 つのモデルに分類した．(a) はクライアント端末からオンラインサービスを利用する場合のモデルであり，ユーザはサービス端末を利用せず，ネットワーク上のサーバからサービス情報を取得する．(b) では，ユーザはサービス端末を利用し，ネットワーク上の情報のアクセスはサービス端末が行う．(c) では，サービス情報はネットワーク上のサーバに加え，サービス端末ごとに固有の情報が保持される．そのため，ユーザがサービス端末を利用する際にはサービス端末で管理する情報がアクセスされる．(d) では，サービス情報はネットワーク上ではなく，サービス端末上で管理される．

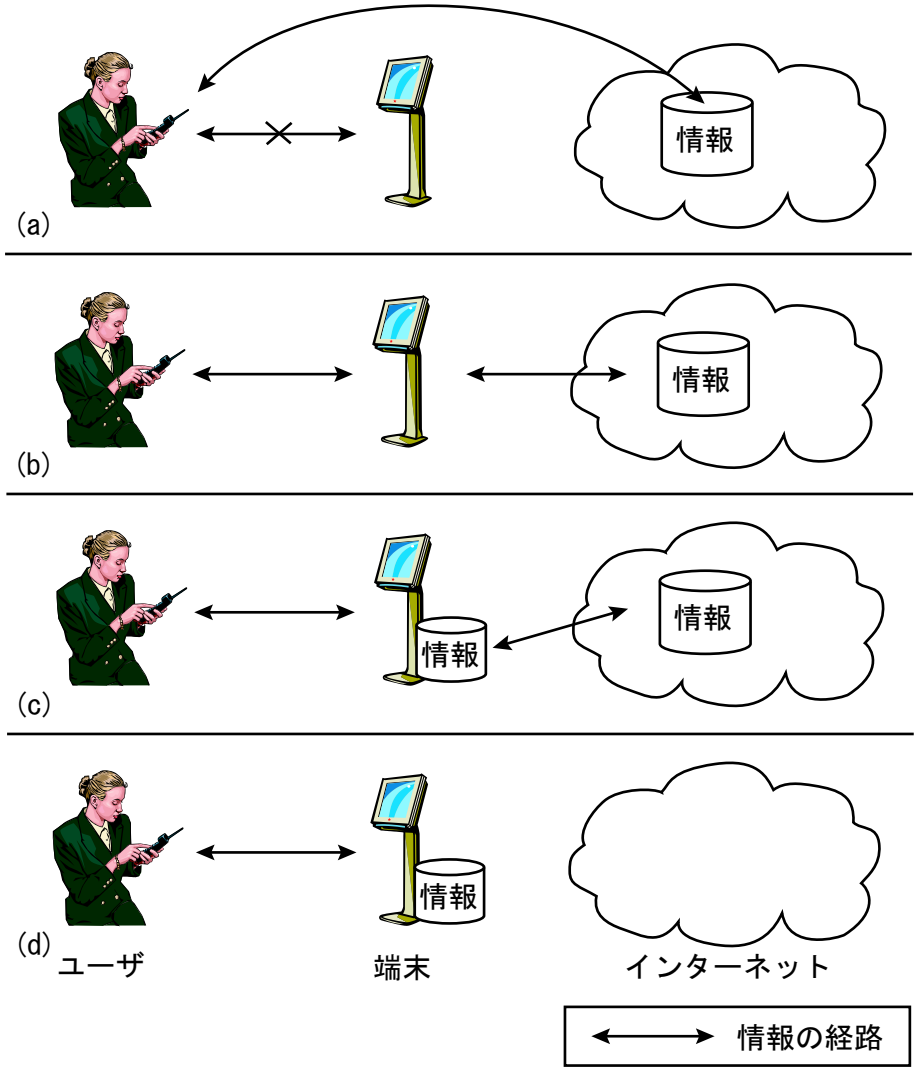

図 2.4: サービス利用形態のモデル

(a) では，ユーザはクライアント端末からネットワーク上のサービス情報をアクセスするため，ユーザの位置情報はユーザあるいはクライアント端末が把握する必要がある．そ

れに対して，その他のモデルではサービス端末がユーザの位置情報を把握しサービス情報を提供する際に利用する．また，サービス情報が管理される場所はモデルによって異なる．1 つのサーバで集中管理した場合は，全ての情報が得られ，分散管理した場合は局所的な情報が得られる．本研究では，ユーザがサービス端末を利用することを想定するため，(b)，(c) および (d) のモデルを考慮する．

2.5 本章のまとめ

本章では，公共空間でサービスを提供する際に考慮する必要がある計算機利用環境，それらの環境を実現するために必要となる無線通信と，提供されているサービスについて検討した．現在の公共空間における計算機利用環境では，ユーザが実際に利用できる計算機資源は乏しく，持ち歩く必要のある機器が多い．また，ネットワークインフラを構築するための技術は存在するが，それらを利用したサービスはホットスポットサービスや携帯電話以外は数少ない．

SmartTerminal を構築するためには，ユーザの計算機利用を支援するために必要なネットワークインフラとサービスを考慮する必要がある．次章において，本章で述べた問題点と課題を基に SmartTerminal のモデルを検討する．

第3章　情報端末 SmartTerminal

本章では，現在の計算機利用環境やネットワークインフラの問題点や課題をとらえ，ユーザに通信，情報および機器機能を提供する情報端末 SmartTerminal のモデルおよびその機能要件を述べる．次に，それらの機能要件を満たすために必要な基盤機構を提案する．

3.1 SmartTerminal

本研究では，既存のホットスポットサービスと携帯端末の問題点に着目し，街中など公共な空間においてユーザの計算機利用活動を支援する情報端末 SmartTerminal を提案する．第 1.1 節で述べたように，現在の計算機利用環境には以下の制約がある．

携帯電話の入出力機能の制約

計算機が人々の生活のあらゆる場面で利用される 1 例として携帯電話がある．携帯電話は小型化して持ち歩きが容易になり，子供から高齢者にまで普及した．またデータ量は増加し，ユーザの要求は複雑になった．携帯電話が出現した当初，扱うことのできる文字は数字や記号に限定された．携帯電話の処理能力や記憶装置が進歩し，ひらがなや漢字などの文字だけでなく，画像や動画も扱えるようになり，利便性は高まった．しかし，携帯電話は電子メール，音楽再生，動画送受信など利用方法が多様化したため，小型の表示など携帯電話の入出力機能に制限されてしまう．

機器持ち運びの煩雑性

機器の小型化や無線化が進むにつれ，計算機や機器を持ち歩き移動中に利用するユーザが増加した．また，インターネットに接続可能な場所を提供するサービスが出現した．しかし，ノートパソコンやその周辺機器などユーザが持ち歩く必要のある機器が多ければ多いほどユーザの負担が大きい．さらに，特定の場所に行かなければインターネットを利用できない．ユーザはあらかじめサービスの場所を知り，さらにその場所に行く必要があるという物理的な制約が大きい．

サービスの解離

インターネットを利用することにより，さまざまな機器やサービスをネットワークで接続できる．インターネット上の一意のアドレスを指定すれば世界中のどこからでもサービスを受けられるため，企業や自治体はオンラインで商品の販売や情報を発信するために利用している．街中でも，インターネットに接続された端末や，運行状況などリアルタイムで情報を発信するサービスが提供されるようになった．しかし，物理的にユーザの目の前にあるサービスであるにも関わらず，インターネットアドレスを指定しなければならない．サービスをインターネット化したため,「ここ」や「その」という状況によって決定されるサービス内容が失われてしまった．

3.1.1 SmartTerminal の目的

SmartTerminal では，上述の問題点を解決してユーザの計算機利用活動を支援することを目的とする．そのために，SmartTerminal は街中の駅やバス停など多くの不特定多

数のユーザが存在する場所において，情報や機器機能を持つサービスを提供する．次に，SmartTerminal に必要な機能要件を 3 つ述べる．

入出力機能の提供

携帯電話の入出力機能には制約があるため，SmartTerminal ではユーザが入出力を行うための機器を提供する．これにより，ユーザが持つクライアント端末の入出力機能が乏しくても，SmartTerminal で提供される入出力機能を利用して計算機利用を継続できる．入出力機能の例として，ディスプレイ，キーボードやスピーカなどの入出力装置があげられる．

周辺環境の利用

機器持ち運びの煩雑性を軽減させるために，ユーザが持ち歩く必要のある機器を最低限に抑え，持ち運びが困難な機器を SmartTerminal で提供する．ホットスポットサービスでは，ユーザがノートパソコンおよび周辺機器を持ち歩く必要があるが，SmartTerminal でインターネット機能を持つ端末やプリンタを提供することで，ユーザが持ち運ばなくとも周辺の環境を利用できる．

近接したサービス端末の利用

近接したサービスの情報を得るために，わざわざネットワーク上のサーバをアクセスする必要があったが，SmartTerminal を利用することでユーザは直接サーバをアクセスする必要がない．ユーザが直接サービスの情報をアクセスした場合は，位置情報が失われる問題があったが，SmartTerminal で位置情報を保持するため，ユーザは位置に依存したサービス情報を受けられる．

3.1.2 SmartTerminal の特徴

SmartTerminal は街中など公共空間に多数設置され，不特定多数のユーザに通信，情報と機器機能を提供する．端末上で動作するサービスは，複数のサービス提供者によって提供される．公共空間に設置されるため，ユーザはサービスを共有して利用する．SmartTerminal の概要図を図 3.1 に示す．本節では，SmartTerminal を構築するために考慮する必要のある特徴を検討する．

提供方法

SmartTerminal は，サービス端末の提供者と端末上で動作するサービスの提供者によって構築される．サービス端末の提供者は最低限の機能として，ネットワークインフラへの

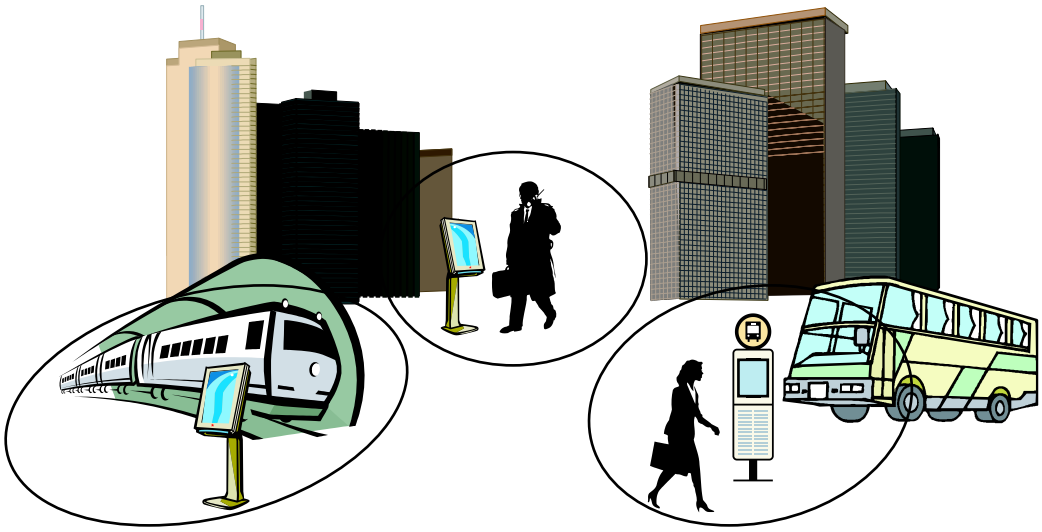

図 3.1: SmartTerminal

接続を可能にする．さらに，ディスプレイやキーボードなどの入出力装置の設置と，複数のサービスに共通する認証や課金などの枠組みを提供する．サービスの提供者は，付加的なサービスをサービス端末に導入する．例えば，駅の近くに設置された SmartTerminal には電車の時刻表サービスが提供される．サービス端末は 1 提供者によって設置されるが，端末上で動作するサービスは複数の提供者が導入できる．例えば，先ほど述べた端末には電車の時刻表が導入されるが，1 社だけでなく異なる複数の企業によってバスの時刻表が導入される場合もある．

配置方法

ユーザが場所に依存せず計算機利用活動を行うためには，街中に SmartTerminal を多数設置してサービスを提供する必要がある．図 3.2 に SmartTerminal の配置方法を 2 つ示す．配置方法の 1 つは公衆電話に用いられている方法である．公衆電話は，駅や街角など人の往来が多い場所に設置された電気通信サービスである．配置されてる数は，人が集中し活動する駅や街中ほど多く，住宅街など居住空間ほど数が少ない．2 つ目は電柱や携帯の基地局などに用いられてる方法である．これらはある一定の間隔で設置され，エリアを満遍なくカバーする．ユーザは SmartTerminal から一定の距離範囲内にいればサービスを利用できるため，無線基地局のように一定間隔で設置することもできる．しかし，サービスを同時に利用できるユーザに制限があり，ユーザが利用する機器によっては目の前にいなければならないため，公衆電話のように人が多く集まる空間では複数設置し，端末間で電波が干渉しないように考慮する必要がある．

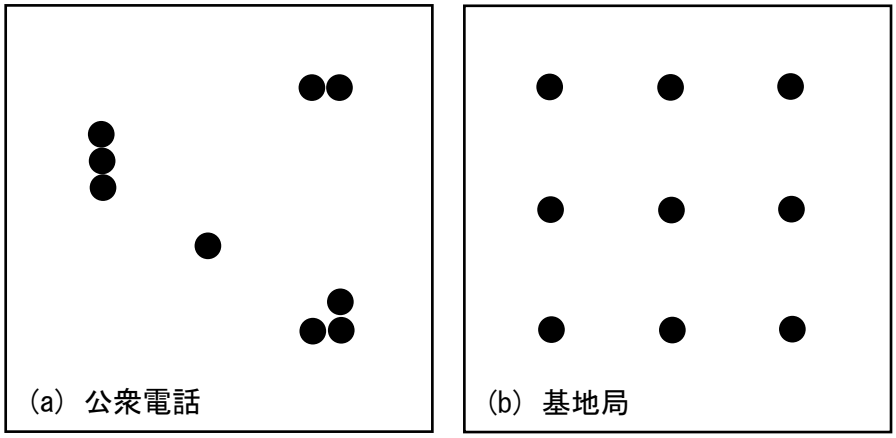

図 3.2: SmartTerminal の配置

共有方法

SmartTerminal では不特定多数のユーザを想定するため，全てのユーザが公平にサービスを利用できるように複数ユーザ間でサービスの共有を行う必要がある．サービスの共有を行うためには 2 つの方法があり，図 3.3 に示す．左は公衆電話の利用モデルであり，一度に 1 人のユーザのみサービスを利用できる．使用を終えたら次の人と交代することで，サービスの共有が行われる．同時に複数のユーザがサービスを受けるためには，複数の端末を用意する必要がある．右は携帯電話の基地局のように同時に複数のユーザが単一のサービスを利用する方法である．1 つの端末を複数のユーザで共有するため，ディスプレイなどの機器を複数ユーザ間で共有可能にする仕組みが必要となる．

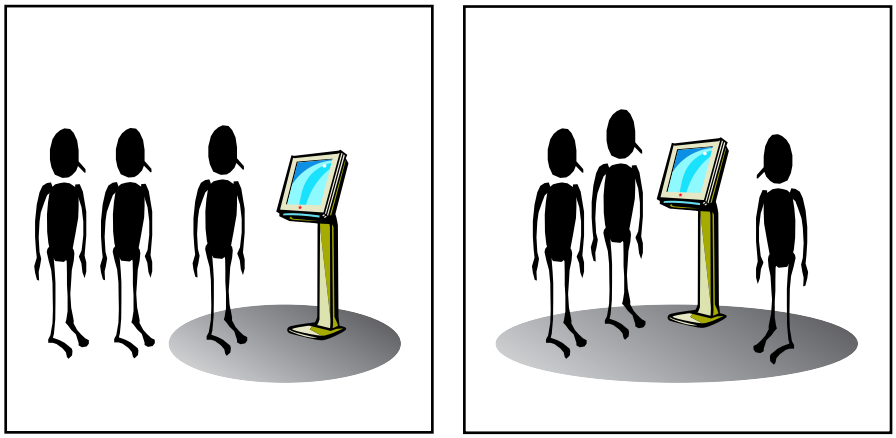

図 3.3: サービスの共有

ユーザと端末の特定

ユーザが SmartTerminal を利用する際に，有料のサービスに対してはユーザの認証や課金が必要となる．また，1 対 1 の通信を行うサービスを利用する際も認証が必要のため，ユーザおよび端末の特定が必要となる．公衆電話と携帯電話のユーザおよび端末の特定の有無を表 3.1 に示す．

表 3.1: 端末と個人の特定

	固有番号	個人の特定	端末とユーザの一致
公衆電話	○	×	×
携帯電話	○	△	×

○：可能 △：場合によって可能 ×：不可能

公衆電話は，各端末を識別するために固有の番号がふられている．しかし，不特定多数のユーザが利用するため個人を特定できず，端末とユーザの 1 対 1 の特定ができない．携帯電話も公衆電話と同様，各端末を識別する固有の番号がふられている．公衆電話と異なり，携帯電話は 1 人のユーザの所有物であるため，携帯電話の固有番号から個人が特定される．しかし，ユーザが携帯電話を他人に貸した場合に端末とユーザの不一致が起こる．つまり必ずしも個人の特定が可能ではない．SmartTerminal では，サービス端末に一意の番号をふる．クライアント端末でユーザの認証を行い特定し，それらを組み合わせてサービス端末とユーザを一致させる．ユーザが利用しているサービス端末が特定できれば，サービス端末に依存せずユーザを指定した通信が可能になる．

通信機能

SmartTerminal では，サービス端末の 1 つの機能として通信機能を提供する．通信機能は，ネットワークインフラの利用およびクライアント端末と無線通信を可能にする．ネットワークインフラに接続することで，ユーザはインターネットサービスを利用できる．インターネットに接続するために ISP を利用するが，ユーザによって契約している ISP が異なるため，ISP を限定せずローミングが可能な機能が必要である．クライアント端末と無線通信を可能にすることで，ユーザはサービス端末とクライアント端末両方からサービスを利用できる．

情報機能

インターネットを利用してオンラインでサービスの情報を提供するサービスが多数存在する．例えば，交通機関はインターネットを利用して運行状況や渋滞情報をリアルタイムで発信する．また，従来は役所や店を訪れなければ受けられなかったサービスがオンラインで利用可能になった．情報機能とは，このようにサービスの情報を取得する手段を呼ぶ．

機器機能

公共空間では，機器を提供するサービスは数少ない．SmartTerminal のような情報端末で機器機能を提供するものはあるが，これらは端末上で提供されるサービスの情報を表示や印刷するためにしか使えず，機器利用がサービスに制限されている．ユーザが計算機活動を行う際には，個人の書類の印刷など，利用目的が既存のサービスの形態とは一致しない．そのため，SmartTerminal ではユーザが自由に使えるディスプレイ，印刷などの機器機能が必要となる．

既存サービスの SmartTerminal 化

SmartTerminal は駅やバス停など街中のあらゆる場所に設置し，ユーザがサービス端末を容易にアクセスできるようにする．導入のコストを最小限にするために，既存のインフラの利用が考えられる．例えば，バス停によってはバスが前の停留所を出発した際に通知するサービスがある．このように既に街中に存在する停留所を，SmartTerminal の基盤機構を導入することで SmartTerminal 化できる．

3.1.3 クライアント端末

ユーザが利用するサービスによっては認証や課金が必要であり，安全に認証を行う方法が必要である．現在，多くのユーザは携帯電話や PDA などの小型携帯端末を持ち歩くため，これらのクライアント端末はユーザを認証する手段として利用できる．

クライアント端末を発見する方法には Pull 型と Push 型の二つあり，それぞれを図 3.4 に示す．Pull 型では，SmartTerminal が常にクライアント端末からの接続要求を待ち受ける．Push 型では，ユーザが常に SmartTerminal からの接続要求を待ち受ける．両者を比較すると，Push 型では SmartTerminal は接続先を発見するために常に接続要求を発行しなければならないが，Pull 型では接続要求はユーザがサービスを受けたい時だけ発行される．SmartTerminal では Pull 型でクライアント端末の発見を行う．

小型携帯端末以外には，IC カードなど ID を有し認証可能な媒体も利用できる．また，ユーザ端末は認証だけではなく，ユーザが SmartTerminal で利用したサービスから情報をダウンロードする先として，あるいはサービスを操作するコントローラとしても利用できる．

3.2 先行事例

公共の空間で通信，情報あるいは機器機能を提供する事例として以下のサービスがあげられる．本節では 3 つの先行事例の特徴と問題点をそれぞれ述べる．

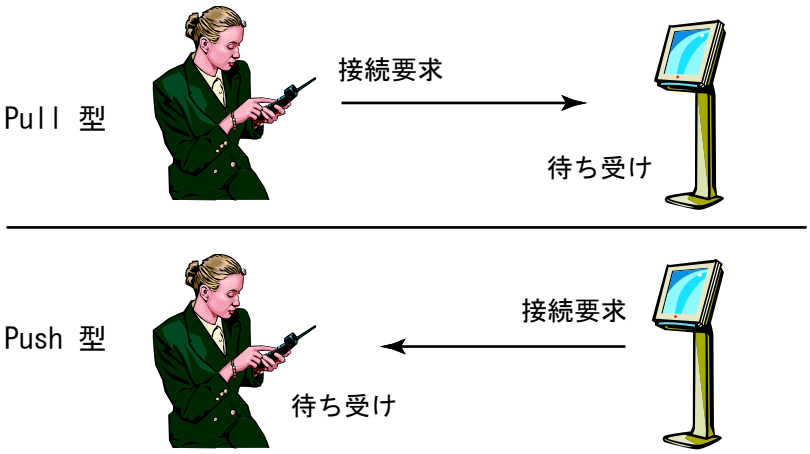

図 3.4: クライアント端末の発見方法

3.2.1 Mobile Internet Services

ネットワークインフラを利用して通信を可能にするサービスとして Mobile Internet Service(以下 MIS)[15] があげられる．MIS では，街角など公共な場所で移動中でも自由に高速にインターネットに接続できる街角無線インターネットサービスを提供する．また，最適なインタフェースやサービスを提供するのではなく，インタフェースやサービスを透過的に利用するためのネットワークインフラ作りを行っている．将来的には日本全国をカバーできる無線基地局の配置を想定している．現段階では範囲は狭めて PC や PDA などを座って利用するなど範囲を限定し，追々移動中に利用する機器に対応する予定である．

MIS は，平成 14 年 4 月から東京都内を中心に無線基地局を設置し，サービスを提供してきたが，同年の 12 月にサービスを休止した．その理由を次に挙げる．

- サービスエリアによって基地局の稼働率に極端な差が生じ，顧客が満足できる形でサービスを継続することが困難になった
- それ以外にも他社と比較して料金が高い
- セキュリティ確保のための専用ドライバの利用など設定が煩雑

あらゆる場所からでもインターネットを利用できるネットワークインフラを構築するには以上の問題点を解決する必要がある．Smart Terminal を構築する上でもネットワークインフラは重要な要素であり，このような運営上の問題を解決する必要がある．

3.2.2 情報 KIOSK 端末

情報 KIOSK 端末 Foobio[16] は，ブロードバンドネットワーク上でコンテンツ流通を行う街頭端末である．駅や CD 販売店などに設置され，PDA など小型携帯端末に音楽や映像といったコンテンツをダウンロードするサービスを提供している．また，デジタルカメ

ラで撮影したデジタルフォトの印刷も可能である．電子政府の取り組みとして，行政手続きを行う情報 KIOSK 端末もある．この端末を利用して時間外の住民票の写し，印鑑登録などの行政サービスが可能になる．情報 KIOSK 端末の製品例を図 3.5 に示す．

図 3.5: 情報 KIOSK 端末 (IBM)

これらのサービスは，マルチメディアコンテンツなど情報だけでなく，印刷などの機器機能も提供する．SmartTerminal でも，これらの情報 KIOSK 端末と同様に情報や機器機能を備えた情報端末を想定する．しかし，これらの情報端末は，サービスに合わせて情報端末が構築されるため，存在する情報端末にサービスを追加できる SmartTerminal のモデルと構築方法が異なる．

3.2.3 Cmode

情報端末と携帯電話を用いたサービスの例として Cmode[17] があり，図 3.6 に示す．Cmode は新型情報端末型自動販売機であり，携帯電話の i モードサービスと連動して利用される．ユーザは携帯電話を利用して，キャッシュレスで Cmode からジュースを購入できる．他には地図やチケットなどの情報をディスプレイ，プリンタやスピーカから出力できる．購入の際に認証や課金は i モードの端末によって行われる．購入者が携帯電話をかざすと個人認証コードが読み込まれ，認証が行われる．

Cmode は従来の自動販売機としての機能以外に，地域情報や機器機能を提供する．Smart-Terminal のモデルでは，利用可能な携帯端末に特に制限はなく，複数の通信手段，情報や機器とともに，それらを提供するための基盤機構を提供する．

図 3.6: Cmode 新型情報端末型自動販売機 (日本コカ・コーラ，エヌ・ティ・ティ・ドコモ，伊藤忠商事)

3.3 本章のまとめ

本章では，SmartTerminal の目的を述べ，目的を実現する SmartTerminal のモデルを検討した．SmartTerminal は，駅などの公共空間にてユーザにさまざまなサービスを提供する．また端末上では多種多様なサービスが複数のベンダーによって提供される．

このような SmartTerminal を実現するために，多種多様なサービスを提供し，それらを管理するための基盤機構が必要となる．次章で SmartTerminal の基盤機構である SmartTerminal Framework の設計を述べる．

第4章　SmartTerminal システムの設計

本章では，本研究で提案する情報端末 SmartTerminal を構築する基盤機構でなる SmartTerminal Framework の設計方針および各構成要素の設計を述べる．SmartTerminal Framework では機能ごとにサービスのモジュール化を行い，統一されたインタフェースを提供する．また，モジュール化されたサービスはSmartTerminal Framework で提供するマネージャーによって管理される．

4.1 SmartTerminal のシステム概要

本研究では，通信，情報および機器機能を提供する情報端末 SmartTerminal を構築するための基盤機構の実装を行った．SmartTerminal はネットワークインフラ，SmartTerminal Framework および SmartTerminal アプリケーションの 3 部分から構成される．ネットワークインフラは，インターネット接続など通信を可能にする．SmartTerminal Framework では実装されたサービスモジュールやクライアント間の通信モジュールを管理する．また，SmartTerminal アプリケーションは実際にユーザが利用するサービスであり，SmartTerminal Framework で実装されたサービスおよび通信モジュールを組み合わせて構成される．

SmartTerminal で提供されるサービスは，そのサービス端末の位置や管理者によって異なる．その様子を図 4.1 に示す．全ての SmartTerminal で利用可能なサービスを提供するためには，SmartTerminal Framework で提供する統一されたインタフェースを実装しなければならない．次に SmartTerminal の基本構成をハードウェアとソフトウェアに分けて述べる．

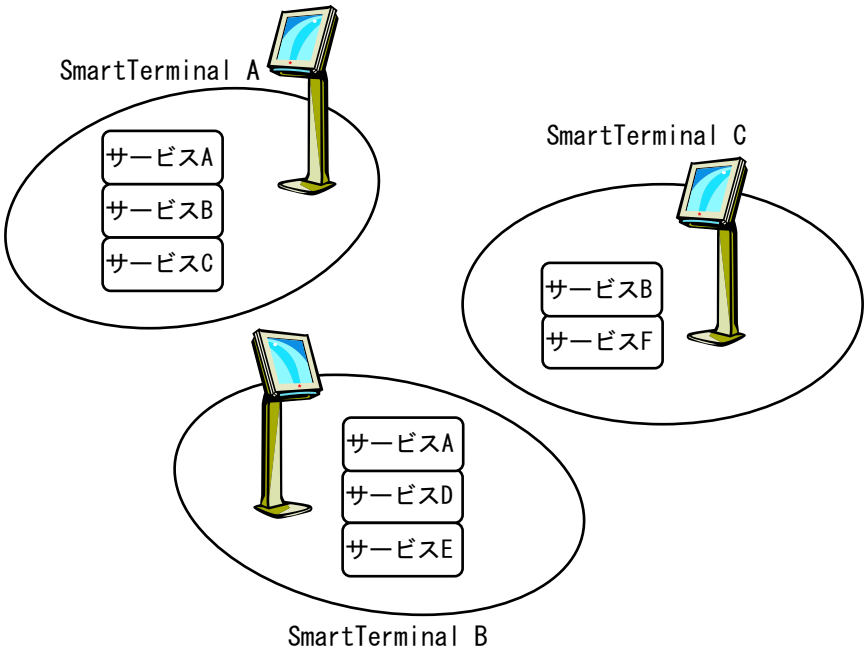

図 4.1: SmartTerminal によって異なるサービスの提供

ハードウェア構成

SmartTerminal のサービス端末のハードウェア構成は，提供されている機器機能，クライアント端末間の通信に必要な通信媒体，外界の情報を取得するために利用するセンサ群およびユーザが持ち歩く端末となっている．機器機能の例としては，ディスプレイ，キー

ボードやプリンタなどの出力装置があげられる．また，通信媒体としては Bluetooth や赤外線，センサ群にはユーザやサービス端末の位置を取得するセンサなどがある．ユーザが利用するクライアント端末としては携帯電話などの携帯端末があり，ユーザが携帯端末を持たない場合は認証が可能な IC カードなどが想定される．

ソフトウェア構成

SmartTerminal のソフトウェア構成は，端末上でユーザに提供されるアプリケーションサービス，SmartTerminal Framework を用いて実装したサービス端末の基盤ソフトウェアやモジュール，および機器が動作するために必要なソフトウェアである．基盤ソフトウェアはサービス端末上で提供されるモジュールを統括して管理し，サービス端末を運営するために用いられる．

4.2 SmartTerminal Framework の設計方針

SmartTerminal のサービス端末は，街中に複数設置されることが想定される．それぞれのサービス端末で提供されるサービスのモジュールは異なり，必要に応じてモジュールの追加や変更が行われる．そのため，不特定多数のサービス提供者が SmartTerminal Framework を用いて様々なモジュールを実装する．以下に SmartTerminal Framework の設計方針を示す．

4.2.1 インタフェースの統一

SmartTerminal に導入されるサービスは場所や状況によって異なるため，あらかじめ知ることができない．それぞれのサービスが独自のインタフェースで実装されてしまうと，SmartTerminal でサービスごとの差を吸収する必要がある．しかし，サービスごとにインタフェースを追加するのでは，管理者にとって付加が高く，汎用性が失われてしまう．そのため，SmartTerminal Framework ではサービスや通信のインタフェースを定義し，サービス提供者はそれらのインタフェースを利用してサービスを実装する．すべてのサービスが統一されたインタフェースを持つため，サービスの開始や停止などの操作が透過的に行える．また，サービスごとにインタフェースを追加する必要がないため，導入のコストが軽減される．

4.2.2 サービスのモジュール化

多くのサービスは複数のサービスを組み合わせて構成される．例えば，バスの運行状況サービスは交通情報サービス，位置情報サービス，バスの時刻表サービスなど複数の単機能を提供するサービスから構成される．また，電車の運行状況サービスもバス同様に位置

情報サービスを必要とする．このように，同一機能を必要とするサービスは複数存在する．それぞれのサービスが全ての機能を実装した場合，同一機能が複数実装され，重複してしまう．SmartTerminal では，サービスの重複を避け，より多くのサービスを提供するためにサービスのモジュール化を行い共有する．SmartTerminal アプリケーションは複数の単機能のモジュールによって構成される．モジュールの組合わせの例を図 4.2 に示す．

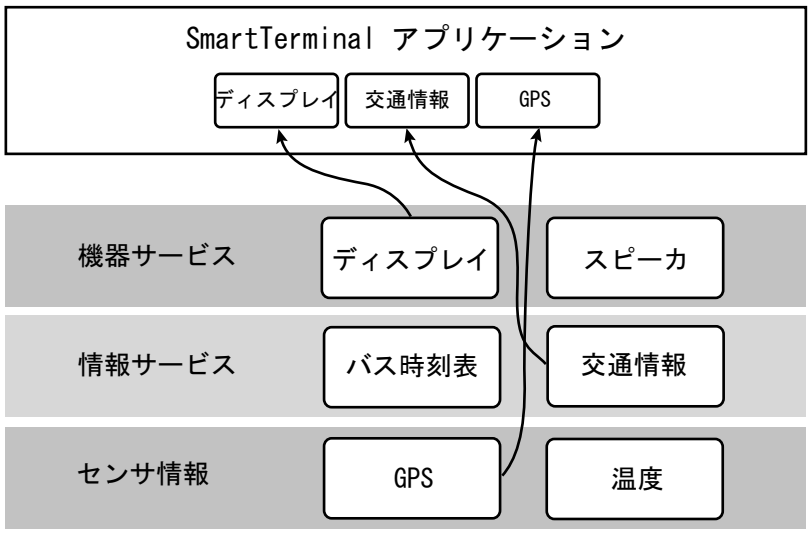

図 4.2: モジュールの組み合わせ例

4.2.3 モジュールの組み合わせ

サービスのモジュール化を行うことにより，SmartTerminal では細かい粒度でサービスを提供できる．位置情報やクライアント認証などのモジュールは利用するアプリケーションが多いため，新しいアプリケーションを追加する際は足りないモジュールのみを拡張すれば良い．例えば， SmartTerminal にバス運行状況サービスが存在し，電車運行状況サービスを追加したい場合，不足している電車の時刻表サービスのモジュールと電車全般の運行状況サービスのモジュールのみを拡張すれば新たに電車運行状況サービスを提供できる．電車アプリケーションを追加する場合のモジュール追加例を図 4.3 に示す．既に存在するモジュールを新たに実装する必要がないため，サービス提供者の負担は軽減され，モジュールの再利用性が高まる．

4.2.4 環境適応性

SmartTerminal では，必要に応じてモジュールが追加および削除されるなどしてサービスが変化する．そのため，サービス端末上で利用可能なモジュールを静的に記述してしまうと，モジュールを変更する度に記述も変更する必要がある．サービス端末が街中に多数

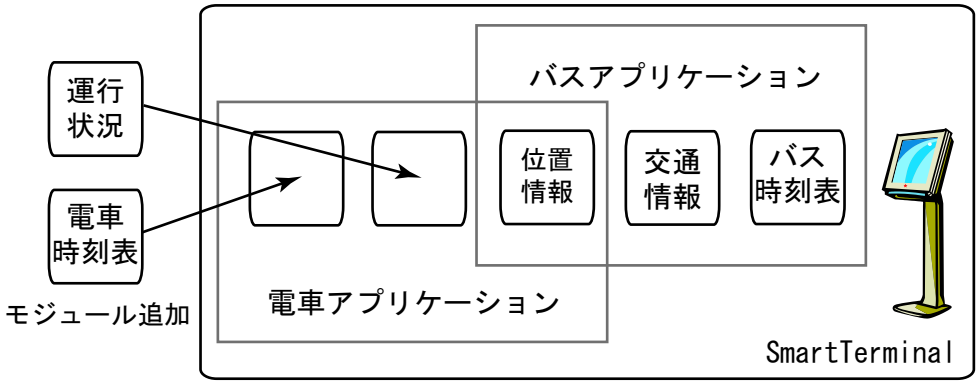

図 4.3: モジュール組み合わせの例

設置された場合，1 端末の保守にかかる負担を最小限に抑える必要がある．サービス端末自身がサービスとモジュールの変更に適応可能であれば，管理者の負担は軽減される．そこで SmartTerminal では，サービス端末上で提供されているモジュールを静的に記述せず，必要に応じて調べる．

4.3 SmartTerminal Framework

第 4.2 節で示した設計方針に従って SmartTerminal Framework の設計を行った．SmartTerminal Framework では，SmartTerminal のサービス端末に必要となる基盤機構を構築するための API を提供する．図 4.4 に SmartTerminal Framework のうちサービスを管理する機構の構成図を示す．

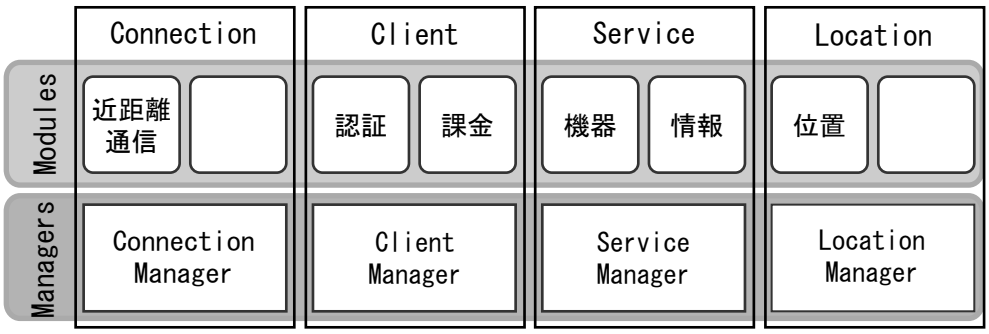

図 4.4: SmartTerminal Framework のサービス管理部分

SmartTerminal Framework のうち，サービスを扱う部分は 2 つの方法で切り分けられる．1 つは，サービス端末で最低限必要な Connection，Service，Client と Location の 4 つの機能でわける方法である．もう 1 つは，Framework を Module 部とそれらのモジュールを管理する Manager 部に分ける方法である．第 4.3.1 項と第 4.3.2 項で SmartTerminal

Framework の Module 部と Manager 部のインタフェースおよび動作手順を示す．また，第 4.3.3 項では SmartTerminal Framework の基盤部分にあたる Core 部を示す．

4.3.1 Module 部

Module 部は Connection，Service，Client と Location の 4 つの機能を提供するサービスモジュールで構成される．各機能をモジュール化するために，機能の抽象化を行い，インタフェースを定義した．

モジュールとは，それぞれ通信や認証など単一の機能あるいはサービスを提供するものである．これらのモジュールをまとめて Module 部と呼ぶ．SmartTerminal を構築するために，提供するサービス (Service Module)，サービスを利用するための仕組み (Client Module)，ユーザ間の通信の支援 (Connection Module) およびサービス端末とユーザの位置を把握する手段 (Location Module) が必要である．

各モジュールは複数のユーザやサービスに利用される．そのため，ユーザやサービスからモジュールの利用要求が来た場合，それぞれに新しいセッションを割り当てる．モジュールを利用し終えるとセッションも終了される．

各モジュールは情報として，自分自身が所属するマネージャの名前，自分自身の名前と ID およびセッションの ID を保持する．モジュールのインタフェースには 2 種類ある．1 つ目は全てのモジュールに共通するインタフェースであり，主にモジュールの保守管理を行うために用意される．これらは SmartTerminal の管理者によって実行される．2 つ目はセッション管理およびサービス機能特有のインタフェースであり，モジュールが提供するサービスを利用するためのものである．各モジュールは両方のインタフェースを実装する．

次に，全てのモジュールが実装するインタフェースを示す．

- **モジュールの初期化 (Init)** モジュールを追加する，あるいは停止していたモジュールを再開する時に初期化を行う．また，必要に応じてモジュール開始に必要な定義ファイルを読み込み設定を行う．
- **モジュールの開始 (Activate)** モジュールを開始する．
- **モジュールの終了 (Terminate)** モジュールを停止する．
- **モジュールの状態取得 (Status)** モジュールは動作中，エラー，アイドルなどの状態を取得する．
- **モジュールのセッション数 (NumberOfSessions)** 動作中のモジュールに対して，モジュールが持つセッションの数を調る．

次に，各モジュールの説明と特有のインタフェースを述べる．

Connection Module

Connection Module は，SmartTerminal のサービス端末とクライアント端末間の近距離無線通信を可能にするモジュールである．クライアント端末とは，ユーザの携帯電話や PDA などの端末を指す．SmartTerminal 端末をクライアント端末と併せて利用したい場合に Connection Module を用いて通信を確立する必要がある．Connection Module の役割は近距離無線を利用した通信の支援であり，ユーザが直接 SmartTerminal を操作する場合には動作しない．近距離無線通信方法としては，Bluetooth や赤外線があるが，実際に利用される通信方法はクライアント端末に備わっている機能に依存する．そのため，SmartTerminal では複数クライアントと通信方法に対応できるように，複数の Connection Module を提供し，それぞれ同時に複数セッションを持てるようにする．

以下に，Connection Module に特有のインタフェースを示す．

- **端末間の通信の確立 (Open)** SmartTerminal はクライアント端末の要求を受取り，通信を確立する．
- **クライアント端末の発見 (Run)** クライアント端末の接続要求を待つ．
- **端末間のデータの送受信 (Read, Write)** SmartTerminal とクライアント端末間を無線でデータの送受信を行うために，Connection Module でデータ送信および受信インタフェースを実装する．
- **暗号化 (Encrypt, Decrypt)** 無線通信を利用した場合，盗聴やなりすましが行われる危険性がある．個人情報や認証鍵が交換されるため，データの暗号化を行いセキュリティを保証する必要がある．そのため，送受信するデータの暗号化および解読を行う．
- **通信の有無の判断 (isConnected)** SmartTerminal とクライアント端末間の通信が有効であることを調べる．
- **通信の失効 (Expire)** SmartTerminal は複数のユーザが同時に利用することを想定しているが，SmartTerminal に同時に接続可能な人数はあらかじめ定められており，サービス資源にも限度がある．通信は，クライアント端末が通信が可能な範囲から出る，ユーザが終了する，あるいは SmartTerminal が自動的に切断する 3 つの場合に終了される．しかし，ユーザが明示的に通信を終了しなければならない場合，終了のし忘れにより資源が解放されないままとなる．資源の枯渇を避けるために，SmartTerminal では利用されていない通信に有効期限をつけ，期限が切れたら自動的に通信を切断する．
- **通信の終了 (Close)** SmartTerminal とクライアント端末間の通信を終了し，資源を解放する．

次に Connection Module の動作手順を示す．動作は通信開始時，サービスの利用時と通信の停止時の 3 段階に分けられる．まず，Connection Module は常にクライアント端末

からの要求を待ち受ける．接続要求を受け付けると通信が確立され SmartTerminal とクライアント端末間のセッションが開始される．サービス利用時に，Connection Module は 2 端末間でデータの送受信を行う．この際は，個人情報が他人に盗まれないように暗号化が行われる．最後に，通信停止時にセッションおよび通信を終了する．

Service Module

Service Module は，情報および機器機能を実装したモジュールである．機器機能としてはディスプレイ，スピーカ，キーボード，プリンタなどの入出力装置があり，Service Module ではそれらの機器を操作するソフトウェアが実装される．情報機能としては，各種時刻表，交通情報，地図情報，周辺情報，マルチメディアなどを提供するサービスがあり，Service Module として実装する際にはそれらの情報を閲覧可能にする．

Service Module は以下のインタフェースを持つ．それぞれセッションを操作するインタフェースとサービスを操作するためのインタフェースである．

- **セッションの開始 (Start)** ユーザあるいはサービスが要求したサービスのセッションを開始する．
- **セッションの中断 (Suspend)** ユーザやサービスは利用中のサービスを中断することができる．
- **セッションの再開 (Resume)** ユーザやサービスは中断したセッションを再開できる．
- **セッションの終了 (Stop)** セッションを終了する．
- **命令の実行 (Execute)** ユーザが選択した項目を実行する．
- **サービスのダウンロード (Download)** ユーザが情報をクライアント端末にダウンロードする際に実行される．
- **サービスの共有 (Share)** サービスの共有を行う．
- **メニューの取得 (GetMenu)** サービスのメニュー一覧を要求する．
- **共有方法の取得 (GetSharingMethod)** サービスの共有方法を要求する．

Client Module

Client Module は，ユーザの認証やサービスの課金を実装したモジュールである．これらの機能は通信の方式に依存せず，多くのサービスが利用する．汎用性が高いため，これらの機能は通信とサービスとは独立して Client Module で提供する．Client Module の例としてユーザ認証，ユーザの個人情報の取得，ユーザのサービス利用状況，サービスの課金があげられる．実装されるインタフェースはモジュールが提供する機能によって異なる

が，ユーザ間のセッションを操作する部分は共通する．セッションに関するインタフェースを以下にあげる．

- **セッションの開始 (Start)** ユーザあるいはサービスモジュールのセッションを開始する．
- **セッションの中断 (Suspend)** ユーザやサービスがモジュールを利用する場合は，それぞれセッションを中断できる．
- **セッションの再開 (Resume)** ユーザとサービスは中断したそれぞれのセッションを再開できる．
- **セッションの終了 (Stop)** ユーザあるいはサービスがセッションを終了．
- **命令の実行 (Execute)** それぞれの Client Module は特化した機能を持つ．例えば，ユーザ認証モジュールはユーザの認証の実行，個人情報モジュールは情報の取得，サービスの課金は課金の実行である．各 Client Module はこのインタフェースに特化した機能の実行をあてる．

Location Module

SmartTerminal は街中に複数設置されるため，サービスの中には位置に依存した機能や情報を提供するため，位置情報が必要になる．例えば，周辺情報サービスは，端末の位置情報を利用してユーザに提示する情報を絞る．また，サービスを提供する上でユーザのサービス端末からの正確な距離を知る必要のあるサービスがあった場合に，距離を測定できる Location Module が必要になる．このように，位置情報を利用するサービスは多数存在する．

LocationModule では，位置情報を要求されると，要求に応じて位置データを加工し，返答するという単一の機能を提供する．そのため，Location Module は他のモジュールと異なり，セッションは存在せず，1 つのインタフェースのみ提供される．例えば，サービス端末の住所を返す Location Module は，所在地を取得するインタフェースを持つ．また，サービス端末とクライアント端末までの距離を測定して返す Location Module では，必要とされている粒度で距離を測定して返答するインタフェースを持つ．

4.3.2 Manager 部

Manager 部は，第 4.3.1 項であげたそれぞれのモジュールを管理する 4 つのマネージャによって構成される．それぞれ，Connection Manager，Service Manager，Client Manager，Location Manager と呼ぶ．マネージャの役割は主に 3 つある．起動や終了などモジュールの操作，各モジュールのセッション管理，およびマネージャ間でのユーザの情報を保持するユーザオブジェクトの受け渡しである．ユーザオブジェクトに関しては第 4.3.3 項の Core 部で説明する．次に，マネージャが提供するインタフェースを検討する．

- **マネージャの初期化 (Init)** 初期化を行う．マネージャが管理する全てのモジュールを初期化しサービスを開始できる状態に準備する．
- **マネージャの開始 (Activate)** 初期化後にマネージャを開始する．
- **マネージャの終了 (Terminate)** マネージャを終了する．
- **マネージャの状態取得 (Status)** 動作中，エラーなどの状態を取得する．
- **マネージャのモジュール数 (NumberOfModules)** 管理するモジュールの数を取得する．
- **動作中のモジュール数 (NumberOfActiveModules)** 管理するモジュールのうち動作中の物の数を取得する．
- **ユーザオブジェクトの受け渡し (Put, Get)** ユーザが異なるマネージャが管理するモジュールを利用したい場合，ユーザオブジェクトがマネージャ間で受け渡される．

各マネージャ間でユーザオブジェクトを受け渡す際の状態遷移を図4.5に示す．このように，SmartTerminalの利用は必ずConnectionManagerから開始され，次にServiceManagerに移る．その後はClientManagerとLocationManagerに必要に応じて実行が移る．

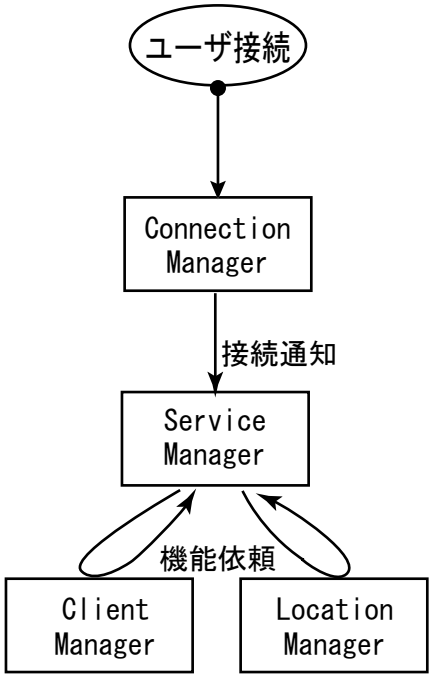

図 4.5: マネージャ間の状態遷移

次にマネージャの動作手順を示す．各マネージャにはユーザオブジェクトを受け渡すためにキューが利用される．あるマネージャが他のマネージャにユーザオブジェクトを受け

渡すには，最初に受取り側マネージャに自身の ID と次に利用するモジュールを指定する．次に，受取り側マネージャのキューにユーザオブジェクトを追加する．受取り側はキューに新たなユーザオブジェクトを発見した場合に，そのオブジェクトを受取り，指定された次のモジュールを実行する．その際にユーザオブジェクトを引数として渡す．

次に，それぞれのマネージャの役割を説明する．

Connection Manager

SmartTerminal では，端末に設置されている無線通信方式の数だけ Connection Module が提供される．Connection Manager では，それぞれの Connection Module を管理し，状態を監視する．

ユーザの SmartTerminal の利用はクライアント端末利用の有無に関わらず Connection Manager から始まる．ある Connection Module がクライアント端末から接続要求を受け付けると，セッションが開始される．Connection Manager では新たなユーザの接続を検知し，そのユーザを表すユーザオブジェクトを生成する．ユーザオブジェクトに通信のセッションを追加し，サービスを受けるために Service Manager に渡される．ユーザがクライアント端末を利用せず，直接 SmartTerminal を操作する場合は，Connection Manager は直接利用を示す Connection Module をユーザオブジェクトに追加し，Client Manager のキューに追加する．

Service Manager

Service Manager は，SmartTerminal 上の情報および機器機能を実装した Service Module を管理する．他のマネージャと同様に，初期化などの操作や，キューにユーザオブジェクトを発見すると指定されたモジュールを実行する．また，一度認証を済ませたユーザオブジェクトはユーザがいつでもサービスを利用できるように Service Manager が保持する．

新規にユーザが SmartTerminal に接続し後，ユーザオブジェクトは必ず Service Manager に渡される．Service Manager は複数のサービスが利用可能な場合は，ユーザに SmartTerminal 上のサービス一覧を送信する．また，利用するために認証が必要の場合は Client Manager に認証を依頼する．その後，ユーザの選択に従って Service Module を実行する．実行された Service Module が他のマネージャが管理するモジュールを必要とした場合に，Service Manager はユーザオブジェクトにそのモジュールの ID を追加し，モジュールを管理するマネージャのキューに追加する．その間，ユーザオブジェクトは，ユーザが利用したサービスの履歴を保存する．

Client Manager

Client Manager は，SmartTerminal 上のユーザ認証やサービスの課金などの Client Module を管理する．各モジュールに対しては，初期化，起動，終了などの基本的な操作を行う．また，ユーザや他のサービスから Client Module の利用を要求された場合，ユーザの

セッションを開始し，サービスを提供する．

Client Manager は常にキューを調べ，新しいユーザオブジェクトの追加を待つ．ユーザオブジェクトを取得した場合，ユーザオブジェクトに指定されたモジュールを実行する．実行後はServiceManager にユーザオブジェクトを渡す．

Location Manager

Location Manager は，SmartTerminal 上の位置情報取得機能を実装した Location Module を管理する．実際には Location Module を直接ユーザが利用することはほとんどなく，Service Module に利用されることが多い．

Location Module の利用者がユーザではなくサービスであっても，利用の際にはユーザオブジェクトに利用するモジュールの情報が追加され，キューに入れられる．Location Manager は指定されたモジュールに要求を渡し，位置情報取得後にもとのサービスモジュールにユーザオブジェクトを返す．

4.3.3 Core 部

Core 部は，SmartTerminal を運営する上で必要となる一般的な機構およびインタフェースを提供する．ユーザオブジェクトやマネージャ間でユーザオブジェクトを受け渡しするための機構は Core 部に含まれる．また，抽象化されたモジュールとマネージャも Core 部に含まれる．

UserObject

ユーザオブジェクトとは，SmartTerminal 内でユーザを表すオブジェクトである．ユーザオブジェクトはユーザプロファイル，利用する通信，サービスなどを含めたセッションの情報を管理を行う．ユーザのオブジェクトを管理するために，ユーザオブジェクトは以下のインタフェースを持つ．

- **プロファイルの取得 (GetProfile)** プロファイルの内容を調べる．
- **プロファイルの追加 (SetProfile)** プロファイルをユーザオブジェクトに設定する．
- **コネクションの取得 (GetConnection)** コネクションのインスタンスを取得する．
- **コネクションの追加 (SetConnection)** コネクションをユーザオブジェクトに設定する．
- **送信先・送信元の設定 (SetDestination, SetSource** ユーザオブジェクトの送信先あるいは送信元を設定する．

- **送信先の取得・設定 (GetDestinationManager,SetDestinationManager)** ユーザオブジェクトを渡す先のマネージャを調べるあるいは設定する.
- **送信宛先の取得・設定 (GetDestinationModule,SetDestinationModule)** ユーザオブジェクトを渡すモジュールを調べるあるいは設定する.
- **送信元の取得・設定 (GetSourceManager,SetSourceManager)** ユーザオブジェクトを送信したマネージャを調べるあるいは設定する.
- **送信元の取得・設定 (GetSourceModule,SetSourceModule)** ユーザオブジェクトを送信したモジュールを調べるあるいは設定する.
- **認証の有無 (isAuthenticated)** 認証の有無を調べる.

4.4 SmartTerminal アプリケーション

SmartTerminal アプリケーションとは，サービス端末上で提供され，ユーザが利用するアプリケーションを指す．それぞれのアプリケーションは端末上の複数の Service や Client のモジュールを組み合わせて構築される.

アプリケーションを構築するために，実装者はアプリケーションに必要な機能を考え，既に SmartTerminal 上で提供されてない部分を新たに実装する．既存のモジュールは，定義されているインタフェースを利用して実装内で利用する．アプリケーションの例は第 6 章であげる.

4.5 本章のまとめ

本章では，SmartTerminal の概要を述べ，SmartTerminal を構築するための SmartTerminal Framework の設計方針および設計を述べた．SmartTerminal Framework では，サービス端末上で提供する各 Connection，Client，Service，Location の機能を実装するためのインタフェースを定義し，提供する．また，それぞれの機能を実装したモジュールを管理するためにマネージャを提供する．本設計では，Connection Manager，Client Manager，Service Manager，Location Manager の 4 つが提供され，次章でそれぞれの実装を述べる.

第5章　SmartTerminal Framework の実装

本章では，前章の設計に基づいて実装したSmartTerminal Framework を詳細に説明する．SmartTerminal Framework では，前章であげた各Connection，Service，Client，Location の機能を実装し，管理するための統一されたインタフェースを提供する．また，それらの機能に共通するCore 部の実装を行い，API として提供する．

5.1 実装概要

SmartTerminal Framework の実装環境として Java2 を用いた．Java は現在様々な機器やアプリケーションの開発で利用されており，基盤に依存せずサービスを提供可能である．また，SmartTerminal では既存の機器やシステムを利用することを想定する．

SmartTerminal Framework の大まかなパッケージ構成を図 5.1 に示す．SmartTerminal Framework は Connection，Service，Client，Location およびこれら 4 つに共通するクラスや SmartTerminal で必要とするクラスを含む Core の 5 つのパッケージによって構成される．

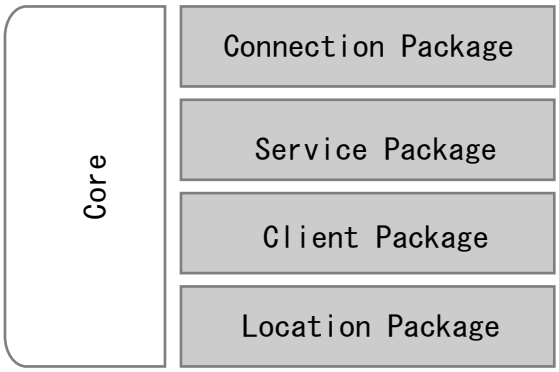

図 5.1: SmartTerminal Framework のパッケージ構成

5.2 Core パッケージの実装

Core パッケージは，SmartTerminal を運営するために必要な API を提供する．本パッケージには，SmartTerminal のメインクラスである SmartTerminal クラス，SmartTerminal の端末情報を取得する STProperty クラス，ユーザを表す UserObject クラス，マネージャ間で UserObject を受け渡すための AccessArea クラスと，Connection，Service，Client，Location の 4 パッケージで実装するマネージャとモジュールの抽象クラスが含まれる．

5.2.1 SmartTerminal クラス

SmartTerminal クラスは，SmartTerminal 端末を操作するためのメインクラスであり，管理者が実行するクラスである．SmartTerminal クラスには，初期化を行う init，端末でサービス提供を開始する start，端末を停止する stop の 3 つのメソッドが提供される．init メソッドでは，SmartTerminal の設定情報を読み込み，それに応じて全マネージャの初期化を行う．

5.2.2 STProperty クラス

各SmartTerminalには，マネージャ，利用可能な通信方式，各種サービスモジュール，それぞれの名前および最大利用可能人数の設定情報を記述したプロパティファイルが用意される．STPropertyクラスはプロパティファイル名を引数で受け取り，指定されたファイルを読み込むことで属性情報を取得する．特に指定のない場合は，ファイル名は**ST.properties**である．ST.propertiesの記述例を図5.2に示す．また，STPropertyクラスの主なメソッドを表5.1に示す．それぞれのメソッドはプロパティの情報を取得するために利用する．

```
NAME=Delta ST
ADDRESS=5322 Endo Fujisawa Kanagawa Japan
MANAGERS=ConnectionManager;ClientManager;ServiceManager;LocationManager
MAX_CLIENT=10

CONNECTION_MODULES=Irda;Bluetooth
CLIENT_MODULES=Accounting;Authentication
SERVICE_MODULES=BusTimetable;Display;Keyboard;Printer
LOCATION_MODULES=GPS
```

図 5.2: ST.properties ファイル記述例

表 5.1: STProperty クラスの主要メソッド一覧

public void getProperties() throws STPropertyNotFoundException;
public String getName();
public String getAddress();
public int getMaxUsers();
public String[] getManagers();
public String[] getConnections();
public String[] getServices();
public String[] getLocations();

5.2.3 UserObject クラス

UserObjectクラスは，SmartTerminalがユーザのセッションおよびサービス情報を管理するために生成されるオブジェクトである．ユーザがSmartTerminalに接続し，認証が行われた後にユーザのプロファイルがUserObjectに設定される．ユーザが利用したサービスやモジュール応じてUserObjectに，通信のセッション，利用しているサービスモジュール，利用履歴などが追加される．また，UserObjectクラスはマネージャ間でユーザの情

報を受け渡す手段として利用される．表 5.2 に UserObject クラスの主なメソッドを示す．これらのメソッドを用いて，ユーザのセッション情報を保持する．

表 5.2: UserObject クラスの主要メソッド一覧

public void setUserInformation(Hashtable UserInfo);
public String getUserName();
public String getInfo(String key);
public String setDestination(String manager, String module);
public String setSource(String manager, String module);
public void setConnection(ConnectionModule connection);
public ConnectionModule getConnection();
public void setService(String service);
public void setAuthenticated(boolean authenticated);
public boolean authenticated();
public void setActive();
public boolean active();

5.2.4 AccessArea クラス

AccessArea クラスは，各マネージャのキューを保持し，マネージャ間で UserObject を受け渡す仲介役として機能する．キューを作成するために，端末上で動作する全てのマネージャを知る必要がある．そのため，AccessArea クラスは SmartTerminal クラスに呼び出される際にマネージャ名と，端末の最大利用人数を取得する．次に AccessArea クラスのフィールド一覧とメソッド一覧をそれぞれ表 5.3 と表 5.4 に示す．

表 5.3: AccessArea クラスのフィールド一覧

型名	フィールド名	説明
short[]	spoolID	Manager の ID
Vector[]	spool	spoolID に対応する Manager の仕事キュー
int	maxUsers	端末の最大利用者数

AccessArea クラスは各マネージャの ID とキューを作成するために変数 spoolID と変数 spool を用いる．各マネージャは他のマネージャに UserObject を渡すために AccessArea クラスの put メソッドを呼び出す．その際に，引数として渡す先のマネージャID を指定する．put メソッドでは，ID を調べ，ID が一致するキューの最後尾に UserObject を追加する．また，マネージャはキューに UserObject が追加されていないかを get メソッドを

```
public synchronized void put(short destID, UserObject user)
   throws UnknownManagerException {

   for (int i=0; i<spoolID.length; i++) {
      if (destID == spoolID[i]) {
         while (spool[i].size() > maxUser) {
            try {
               wait();
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
         spool[i].add(user);
         notifyAll();
      } else if (i == spoolID.length) {
         throw new UnknownManagerException();
      } else {
         continue;
      }
   }
}

public synchronized void get(short sourceID)
   throws UnknownManagerException {

   for (int i=0; i<spoolID.length; i++) {
      if (sourceID == spoolID[i]) {
         while (spool[i].isEmpty() == true) {
            try {
               wait();
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
         UserObject tmp = (UserObject)spool[i].remove(0);
         notifyAll();
         return tmp;
      }
   }

   throw new UnknownManagerException();
}
```

図 5.3: AccessArea のソースの一部

表 5.4: AccessArea クラスのメソッド一覧

public void getManagerIDs(String[] mngrs) throws UnknownManagerException;
public synchronized UserObject get(short sourceID) throws UnknownManagerException;
public synchronized void put(short sourceID, short destID, UserObject user) throws UnknownManagerException;
public synchronized void put(short destID, UserObject user) throws UnknownManagerException;

用いて調べる．UserObject が存在した場合に，先頭のオブジェクトをキューから取り出す．図 5.3 に AccessArea のソースの一部を示す．

5.2.5 Manager クラス

各 Connection，Client，Service，Location 部のそれぞれで実装するマネージャの抽象クラス，Manager クラスは Core パッケージに含まれる．各 Manager クラスは表 5.5 のパラメータ，表 5.6 の主なフィールド，および表 5.7 の主なメソッドを持つ．各マネージャは Manager クラスを継承して実装され，各々のパッケージに含まれる．

表 5.5: Manager クラスのパラメータ一覧

public static final String name	マネージャの名前
public static final short ID	マネージャの ID
public static final short MANAGER_INACTIVE=0	停止中
public static final short MANAGER_ACTIVE=1	動作中
public static final short MANAGER_ERROR=2	エラーの発生

パラメータとしては，名前や ID などマネージャの普遍な基本情報を保持する．それぞれはクラス内のメソッド，あるいは AccessArea クラスのメソッドに利用される．また，各マネージャは現在取り扱っている UserObject およびモジュールのインスタンス持つ．

マネージャは，キューから UserObject のインスタンスを取得すると，UserObject クラスの getDestinationModule メソッドを呼び出して，実行すべきモジュールを調べる．次に UserObject を引数として，モジュールの名前からそのモジュールの exec メソッドを呼び出して実行する．

表 5.6: Manager クラスの主なフィールド一覧

型名	フィールド名	説明
UserObject[]	users	現在扱っている UserObject
Module[]	modules	管理する Module
short	status	現在の状態。パラメータ値を入れる

表 5.7: Manager クラスの主なメソッド一覧

public void init(String[] modules);
public void activate();
public void terminate();
public short getStatus();
public String getName();
public int getNumberOfModules();

5.2.6 Module クラス

各 Connection，Client，Service，Location 部で提供するモジュールの抽象クラス Module クラスもマネージャ同様に Core パッケージで提供する．表 5.8 に主なパラメータ一覧，表 5.9 に主なメソッド一覧を示す．モジュールのパラメータはほぼマネージャと同一であり，モジュールの基本的な情報を保持する．また，前章の設計で述べた以下のメソッドを必ず実装する．

表 5.8: Module クラスのパラメータ一覧

public static final String name	モジュールの名前
public static final short ID	モジュールの ID
public static final short MODULE_INACTIVE=0	停止中
public static final short MODULE_ACTIVE=1	動作中
public static final short MODULE_ERROR=2	エラーの発生

5.2.7 イベントハンドリング

ユーザのセッション管理を行うために，モジュールとマネージャ間でイベントハンドリングを行う．そのために，各パッケージで Event クラスと Listener クラスの継承クラスを実装する．また，それぞれは Module クラスと Manager クラスを継承する全てのクラスによって実装される．ユーザが利用しているモジュールが他のマネージャのモジュールを

表 5.9: Module クラスの主なメソッド一覧

protected void init();
protected void activate();
protected void terminate();
protected short getStatus();
protected int getNumberOfSessions();
public void exec(UserObject user);

必要とした場合，モジュールはイベントに UserObject のインスタンスを挿入して，イベントを発生させる．リスナを実装したマネージャはそのイベントを受け取り，UserObject を取り出す．その様子を図 5.4 に示す．動作手順は以下のようになる．

1. モジュールは UserObject に送信先マネージャの ID とモジュールの名前を追加
2. UserObject を引数に Event を発生させる
3. マネージャはリスナを用いてイベントを受け取る
4. イベントから UserObject を取り出す
5. UserObject の送信先マネージャを調べ，送信先マネージャのキューに追加する

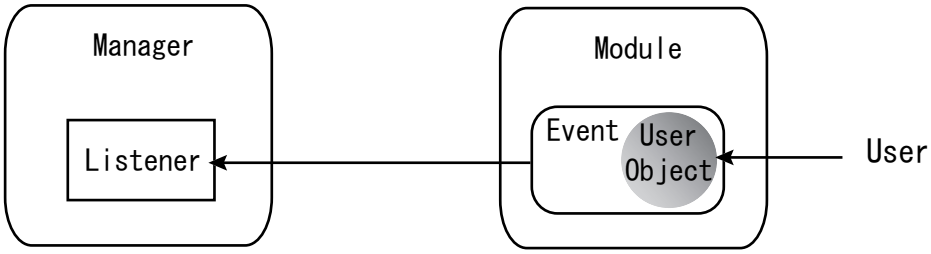

図 5.4: イベントハンドリング

5.2.8 エラーの処理

SmartTerminal Framework では，モジュール，サービスやプロパティファイルが見つからない場合にエラーが発生する．それぞれ Exception クラスを継承する．例えば，マネージャおよびモジュールを初期化する際にマネージャが見つからなければ，UnknownManagerException が発生する．

5.3 Connectionパッケージの実装

Connection パッケージは，SmartTerminal で通信を行うためのクラスを提供する．パッケージは，ConnectionManager，モジュールを実装するためのインタフェースである ConnectionModule クラス，新しいユーザを検知するための ConnectionListener クラスと ConnectionEvent クラス，およびエラー処理を行う UnknownConnectionException クラスから構成される．本節では ConnectionManager クラスおよび ConnectionModule クラスについて述べる．

5.3.1 ConnectionManager クラス

SmartTerminal で提供するクライアント端末間のコネクションを ConnectionManager クラスで管理する．抽象クラスである Manager クラスにて共通なパラメータ，フィールドとメソッドが定義されているが，本実装で利用したパラメータ値を表 5.10 に，フィールドを表 5.11 に示す．

表 5.10: ConnectionManager クラスのパラメータ一覧

public static final String name=”ConnectionManager”	マネージャの名前
public static final short ID=0	マネージャの ID
public static final short MANAGER_INACTIVE=0	停止中
public static final short MANAGER_ACTIVE=1	動作中
public static final short MANAGER_ERROR=2	エラーの発生

表 5.11: ConnectionManager クラスのフィールド一覧

型名	フィールド名	説明
AccessArea	aa	各マネージャのキューを保持
UserObject[]	users	現在扱っている UserObject
Module[]	modules	管理するコネクションのモジュール
String[]	conn	各コネクション名

ConnectionManager クラスは，SmartTerminal クラスに AccessArea のインスタンスと各コネクションの名前を String 配列で受け取る．コネクションの名前を利用して，図 5.5 に示す通り各 ConnectionModule クラスの初期化を行う．

```
public void init(String[] mods) throws UnknownModuleException {
   ClassLoader cl = ClassLoader.getSystemClassLoader();

   for (int i=0; i<mods.length; i++) {
      try {
         modules[i] =
      (ConnectionModule)cl.loadClass(mods[i]).newInstance();
      } catch (ClassNotFoundException ce) {
         throw new UnknownManagerException();
      } catch (InstantiationException ie) {
         ie.printStackTrace();
      } catch (IllegalAccessException iae) {
         iae.printStackTrace();
      }
   }
}
```

図 5.5: コネクションモジュールの初期化

5.3.2 ConnectionModule クラス

ConnectionModule クラスはコネクションモジュールが実装するインタフェースを定義する．SmartTerminal で通信形態を提供するクラスは必ず Module クラスを継承し，ConnectionModule クラスを実装する．表 5.12 に設計を基に実装したメソッドの一部を示す.

表 5.12: コネクションモジュール実装例のメソッド一覧

public void open();
public void run();
public byte[] read();
public void write(byte[] bytes);
public boolean connected();
public void expire();
public void close();

ConnectionModule クラスを実装するモジュールでは，open と close メソッドを利用して入出力用のストリームを用意する．また，read と write メソッドを利用してコネクションに対して読み書きを行う.

5.4 Client パッケージの実装

Client パッケージは，認証など複数のサービスモジュールが利用する機能を提供するモジュールのインタフェースや関連するクラスが含まれる．パッケージは，ClientManager

クラス，クライアントモジュール特有のインタフェースを定義する ClientModule クラス，イベントに関連する ClientListener クラスと ClientEvent クラス，およびエラーを処理する UnknownClientModuleException クラスによって構成される．本章では，ClientManager クラスと ClientModule クラスについて述べる．

5.4.1 ClientManager クラス

SmartTerminal で提供する認証や課金方法のクライアントモジュールは ClientManager クラスで管理する．抽象クラスである Manager クラスで定義されたパラメータのうち，変数 name には”ClientManager”，変数 ID には 1 がそれぞれ値として代入される．また，フィールドは ConnectionManager クラスと同様に，AccessArea クラス，UserObject クラス，Module クラスおよび Module の名前を保持する String が用意される．

5.4.2 ClientModule クラス

ClientModule クラスはクライアントモジュールが実装するインタフェースを定義する．SmartTerminal ではクライアントモジュールとして認証モジュールや課金モジュールが実装される．それらは必ず Module クラスを継承し，ClientModule クラスを実装する．表 5.13 にメソッドの一部を示す．それぞれのメソッドを利用して認証などのセッションを操作する．

表 5.13: クライアントモジュール実装例のメソッド一覧

public void start();
public void suspend();
public void resume();
public void stop();
public void exec(UserObject obj);

5.5 Service パッケージの実装

Service パッケージは，情報や機器機能を提供するサービスモジュールのインタフェースや関連するクラスが含まれる．パッケージは，ServiceManager クラス，サービスモジュール特有のインタフェースを定義する ServiceModule クラス，イベントに関連する ServiceListener クラスと ServiceEvent クラス，およびエラーを処理する UnknownServiceModuleException クラスによって構成される．本章では，ServiceManager クラスと ServiceModule クラスについて述べる．

5.5.1 ServiceManager クラス

ServiceManager クラスでは，SmartTerminal で提供する情報と機器サービスのモジュールを管理する．抽象クラスである Manager クラスで定義されたパラメータのうち，変数 name には”ServiceManager”，変数 ID には 2 がそれぞれ値として代入される．また，フィールドは他のクラスと同様に，AccessArea クラス，UserObject クラス，Module クラスおよび Module の名前を保持する String が用意される．

5.5.2 ServiceModule クラス

ServiceModule クラスはサービスモジュールが実装するインタフェースを定義する．各サービスモジュールは必ず Module クラスを継承し，ServiceModule クラスを実装する．表 5.14 にパラメータの一部，および表 5.15 にメソッドの一部を示す．サービス特有のメソッドとして，情報のダウンロードに利用する download メソッド，機器共有を行うための getSharingMethod および share メソッドがある．download メソッドは，ダウンロードする対象の情報のキーを引数として呼び出す．機器機能の共有を行うためには，最初に getSharingMethod メソッドを呼び出し，値が Synchronous であれば，share メソッドを用いて機能の共有を行う．

表 5.14: サービスモジュールのパラメータ一覧

public static final String name=”ServiceModule”	モジュールの名前
public static final short ID=2	モジュールの ID
public static final short MODULE_INACTIVE=0	停止中
public static final short MODULE_ACTIVE=1	動作中
public static final short MODULE_ERROR=2	エラーの発生
public static final String SHARINGMETHOD=”Synchronous”	共有方法
public static final int SHARINGCAPACITY=10	同時に共有可能な人数

表 5.15: サービスモジュール実装例の一部のメソッド

public void download(String key);
public void share();
public String[] getMenu();
public String getSharingMethod();
public int getSharingCapacity();

5.6 Locationパッケージの実装

Locationパッケージは，位置情報を提供するモジュールのインタフェースや関連するクラスが含まれる．パッケージは，LocationManagerクラス，ロケーションモジュール特有のインタフェースを定義するLocationModuleクラス，イベントに関連するLocationListenerクラスとLocationEventクラス，およびエラーを処理するUnknownLocationModuleExceptionクラスによって構成される．本章では，LocationManagerクラスとLocationModuleクラスについて述べる．

5.6.1 LocationManagerクラス

LocationManagerクラスでは，SmartTerminalで提供する位置情報取得モジュールを管理する．抽象クラスであるManagerクラスで定義されたパラメータのうち，変数nameには”LocationManager”，変数IDには3がそれぞれ値として代入される．また，フィールドは他のクラスと同様に，AccessAreaクラス，UserObjectクラス，Moduleクラスおよび Moduleの名前を保持するStringが用意される．

5.6.2 LocationModuleクラス

LocationModuleクラスはロケーションモジュールが実装する必要があるインタフェースを定義する．実装するクラスは必ずModuleクラスを継承し，LocationModuleクラスを実装する．表5.16にメソッドの一部を示す．getAddressメソッドを利用することでSmartTerminalの位置を取得できる．また，measureメソッドを長さの粒度を指定して呼び出すと，クライアント端末とサービス端末間の距離が測定できる．

表 5.16: ロケーションモジュール実装例のメソッド一覧

public String getAddress();
public int measure(String key);

5.7 本章のまとめ

本章では，前章で述べたSmartTerminal Frameworkの設計に基づいて行った実装について述べた．本研究ではSmartTerminal FrameworkをCore，Connection，Client，Service，Locationの5つのパッケージに分けて実装を行った．それぞれ複数のモジュールとそれらを管理するマネージャより構成される．本章では主にマネージャとモジュールを取り上げ，実装したメソッドやソースプログラムを示した．

第6章　SmartTerminal アプリケーション

本章では，SmartTerminal Framework を用いて実装した SmartTerminal アプリケーションの例として SmartTaxi と SmartBus をあげる．SmartTaxi は，ユーザがタクシーのいない乗り場に着た場合に周辺のタクシーにその旨を通知する．また SmartBus では，ユーザがバス停に近づくと，携帯端末でバスの時刻表や現在の運行状況を取得できる．

6.1 SmartTaxi

SmartTaxi アプリケーションとは，既存のタクシー乗り場を SmartTerminal 化した場合に想定されるアプリケーション例である．現在のタクシー乗り場のモデルでは，タクシー乗り場は人が大勢集まる駅，ショッピングモールやホテルなどに設置される場合が多い．タクシーは通勤時間や終電の時間に多く乗り場に集まる．しかし，タクシーが待っていない時に人が乗り場に来る時がある．乗り場に人が大勢いたとしてもタクシーが一台でもいれば，無線で他のタクシーに連絡をとることが可能だ．しかし，一台もいない場合は，人が待っていることを空車のタクシーが知る手段はない．SmartTaxi では，人がタクシーの待っていない乗り場に並んだ場合に，タクシーを呼ぶサービスを提供できる．

6.1.1 SmartTaxi の実装概要

SmartTaxi を構築するためには，客が端末からタクシーを要求するための単純表示と，要求を受取りサービス端末の位置情報と共に客が待っていることをタクシー会社に通知するサービスモジュールが必要となる．また，位置情報はサービス端末の住所を保持するロケーションモジュールを利用して取得する．

6.1.2 Module の実装

SmartTaxi を構築するためには以下の 2 つのモジュールが必要となる．

- **TaxiService モジュール** 客の要求を受け付けると，タクシー会社にタクシー乗り場に客が待っていることを通知．
- **AddressLocation モジュール** SmartTerminal の位置を取得．

これら 2 つのモジュールを用いて，SmartTaxi を実装した場合の動作手順を次に示す．

1. TaxiService は客から配車要求を受け付けると，start メソッドを実行セッションを開始する．exec メソッドを実行する．
2. UserObject の送信先に LocationManager と AddressModule を，送信元に ServiceManager と TaxiService を設定する．
3. UserObject を用いて ServiceEvent をインスタンス化し，イベントを発生させ，返答を待つ．
4. ServiceManager はイベントを受け取り，送信先マネージャを調べ，LocationManager のキューに追加する．
5. LocationManager はキューから UserObject を取り出し，送信先モジュールを調べ，AddressModule のインスタンスに UserObject を渡す．

6. AddressModule は住所を取得し，UserObject の setInfo メソッドを呼び出して住所を記録する．その際にキーを address，値を住所の文字列とする．

7. AddressModule は送信先を ServiceManager と TaxiService に書き換え，送信元は消す．イベントを発生させ，UserObject を TaxiService に返す．

8. TaxiService は UserObject より住所を取り出し，それをタクシー会社に通知する．

次に，実装した TaxiService モジュールのコードの一部を図 6.1.2 に示す．このように，サービス提供者がサービスを実装する際に，SmartTerminal Framework で提供されるインタフェースの中身を実装し，モジュール間の情報の配送を SmartTerminal Framework に委ねる．

```
public void exec(UserObject user) {
  user.setDestination("LocationManager", "AddressLocation");
  user.setSource(new String(package+"Manager"), name);

  man.serviceRequested(new ServiceEvent(user));

  while(obj.sessionID != sessionID) {
    try {
      wait();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }

  address = user.getInfo("address");

  // call taxi service with address
}
```

図 6.1: TaxiService の exec メソッドのソース

6.2 SmartBus

SmartBus アプリケーションは，バス停を SmartTerminal 化したアプリケーションの例である．現在バス停を利用する際に，以下の状況が考えられる．バス停には時刻表があり，利用者はそれを見て次のバスの時刻を見る．しかし，バス停に列ができている場合，時刻表を確認できないことが多い．また，パスは交通状態に左右されてしまうため，混雑時はバスは時刻表通に到着しない．このような状況を想定して，以下のような SmartBus アプリケーションが考えられる．

- 時刻表を直接確認できない場合，携帯端末に時刻表をダウンロード
- バスの到着予定時刻を携帯端末にダウンロード

6.2.1 SmartBus の実装概要

SmartBus アプリケーションを構築するために，バスの時刻表を提供する情報サービス，客のクライアント端末と通信を行う ConnectionModule，時刻表をユーザに送るサービスモジュール，バスの運行状況を取得できるサービスモジュール，SmartBus 端末の所在をから到着予定時刻を計算するアプリケーションが必要となる．

6.2.2 Module の実装

SmartBus を構築するためには以下の 2 つのモジュールが必要となる．

- **BusService モジュール** バスの時刻表を提供.
- **IrdaConnection モジュール** 赤外線を利用してデータの読み書きを行う.

これら 2 つのモジュールを用いて，SmartBus を実装した場合の動作手順を次に示す.

1. それぞれのマネージャにより BusService および IrdaConnection が初期化され，開始される．客からの時刻表取得要求を待つ.
2. 赤外線がクライアント端末を発見すると，コネクションを開始し，UserObject を生成し，コネクションのインスタンスを UserObject に入れる.
3. UserObject の送信先マネージャに ClientManager を指定し，ClientManager のキューに追加する.
4. ClientManager は UserObject をキューから取り出し，送信先モジュールを調べる．空であるため，送信先モジュールを BusService，送信先マネージャを ServiceManager に変更し，送信元を自分自身の ID にする.
5. ServiceManager は UserObject をキューから取り出し，送信先の BusService のインスタンスに UserObject を渡す.
6. BusService は客にメニューを提示する．メニューには時刻表と運行状況があげられる.
7. 時刻表が選択された場合は BusService は時刻表ファイルを読み込み，データをバイト列に変換し，コネクションのストリームに書き込む.
8. 運行状況が選択されると，SmartTaxi と同じ手順でサービス端末の住所を取得して，位置情報を元にバス会社に運行状況を問い合わせる.

9. 返答を受け取ると，文字列のデータをバイト列に変換して，コネクションのストリームに書き込む.

6.3 本章のまとめ

本章では，SmartTerminal アプリケーションの例として SmartTaxi と SmartBus をあげ，それぞれの概要と実装するモジュールの動作手順および一部のコードを述べた．このように，SmartTerminal アプリケーションは複数のモジュールを実装し，モジュールの組み合わせおよび配送を SmartTerminal Framework で行う.

第7章　SmartTerminal の評価

本章では，本研究で提案した SmartTerminal の評価を示す．評価として，SmartTerminal Framework を用いて構築した SmartTerminalアプリケーションの動作を定量的に評価した．その際に，SmartTerminal の初期化および動作部分の測定を行い，動作を検証した．

7.1 測定環境

測定環境として，SmartTerminal と見立てて一台の計算機を利用した．その計算機の環境を表 7.1 に示す．

表 7.1: 測定環境

CPU	主記憶	OS	JDK
UltraSPARC-II 248MHz × 2	512MB	Solaris 7	HotSpot Client VM 1.3.1

7.2 SmartTerminal 初期化に要する時間

本実装では，SmartTerminal は Connection，Client，Service，および Location の 4 つのマネージャーとそれぞれモジュールを一つずつ実装して初期化に要する時間を測定した．初期化を行う際は，SmartTerminal クラスを利用し，SmartTerminal クラスからそれぞれのマネージャ，それぞれのマネージャは管理するモジュールの初期化を行うため，SmartTerminal を実行してから最後のマネージャの初期化が終了するまでを測定した．この動作を 100 回試行し，平均値をとった．その結果を図 7.1 に示す．

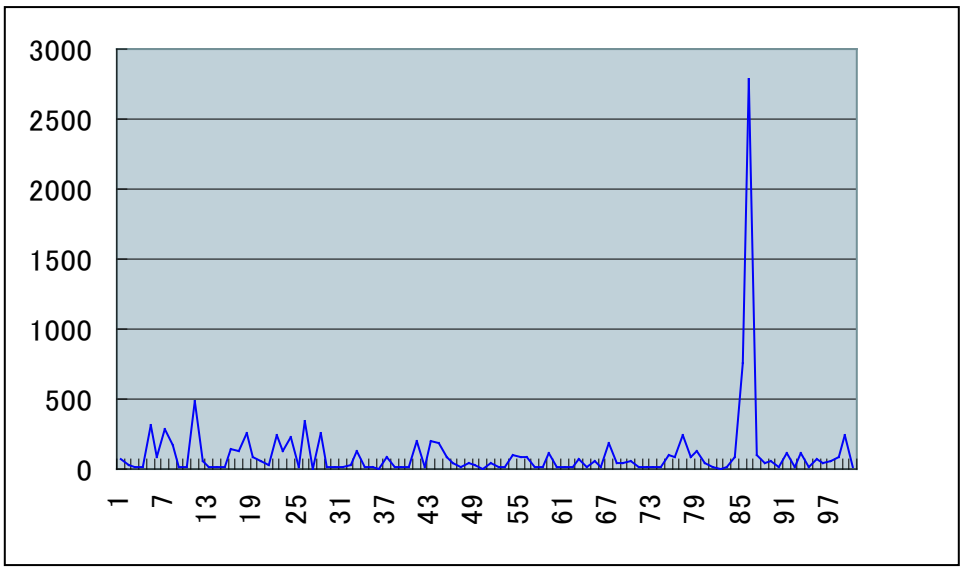

図 7.1: SmartTerminal 初期化に要する時間

測定の結果，SmartTerminal の初期化に要する時間は 108msec であることがわかった．初期化では，主にモジュールのインスタンスを作成している．提供するモジュール数が増加すると，初期化に要する時間も増加することが言えるが，初期化は SmartTerminal を起動する際に 1 回実行するだけであるため，十分であると考えられる．

7.3　本章のまとめ

本章では，SmartTerminal Framework の評価を行った．SmartTerminal Framework のうち，サービス端末の初期化の際に要する時間を定量的に評価した．

第8章　関連研究

本章では，公共空間でサービスを提供するSmartTerminalの関連研究を検証する．第3章では実用化されている先行研究をあげたため，本章では基盤機構を切口として街角情報端末，Saharaおよびone.world，それぞれの特徴を述べ，SmartTerminal Frameworkと比較する．

8.1 街角情報端末

災害対応機能を有する街角情報端末等に関する研究 [18] では，災害発生時に災害情報を流す手段として街角情報端末と携帯電話を利用するシステムの構築を行っている．1998年までに，地域情報を流す街角情報端末の開発を検討していたが，当時は適したインフラ，アクセシビリティや設置台数が問題となった．その代替策として既に街角にあるバス停の案内板など既存のインフラを利用した情報発信手段の開発を行った．近年はアクセシビリティの点で身近な携帯電話が普及したことにより，最寄りの基地局から携帯電話で災害情報を取得する枠組みにシステムが変更された．それまで開発を行っていた街角情報端末は，携帯電話を利用できない人への情報提供手段として利用される．

1998 年当時と比較し，現在では無線やネットワークは街中で普及しており，インフラが整備されつつある．また，情報端末の開発および実用化を行っている企業も多数存在するため，今後街角情報端末のアクセシビリティは良くなると考えられる．携帯電話という単一の機器に依存しているため，今後ネットワークインフラが更に普及した時，インフラに適応するために再びシステムを変更しなければならない．

8.2 Sahara

SAHARA[19] では，複数のサービス提供者がサービスの生成，追加，管理できる包括的な基盤機構の構築を行っている．SAHARA が提案する基盤機構は，サービス資源の導入および管理を行い，コミュニティごとにパフォーマンスや利用の度合いから最適配置を行う．また，複数サービス提供者にわたって，サービスの相互運用および合成を可能にする．この時，サービスの信頼性や動作の検証が必要である．SAHARA を実現することで，例えば海外旅行に携帯電話を持っていくと，携帯電話は旅行先で提供されている通信会社を通してローミングが行え，普段携帯電話で利用している情報サービスは，旅行先の位置情報を元に情報を提供する．

SAHARA では，主にビジネスモデルに着目し，複数サービスプロバイダが提供するサービスを安全に利用するための基盤機構を提案する．しかし，SmartTerminal とは異なり対象とする環境とサービスは幅広く，実現するための基盤機構は複雑になってしまう．また，実現された基盤機構はまだ提供されていない．

8.3 one.world

one.world[20] は，多種多様なサービスが分散されたパーベイシブな環境において，ユーザの継続的な計算機利用を可能にし作業を支援することを目標とする．想定する環境では，サービスやユーザが共に移動可能であり動的に変化する．そのため，環境に反応し，サービスが互いに協調することで機能を提供し続けることが要求される．そのためには，サービス実装者は常に環境に動的に適応するサービスを構築する必要がある．one.world ではパーベイシブ環境で動作するアプリケーションを構築するための単純なプログラミン

グモデルを提供し，環境とサービスに応じてサービスコンポーネントの動的構成を可能にする．サービスの組合わせはイベントハンドラのインポートとエクスポートを宣言することで実現される．

サービスコンポーネント同士が協調作業を行うためには，全てのサービスが one.world のインスタンスを持ち，共有可能でなければならない．環境内の全てのサービスが常に互いに協調動作する必要のある環境においては有益であるが，協調動作やサービスの移動が頻繁には行われない環境においてはシステムを導入するためのコストおよび資源が増加してしまう．

第9章　結論

9.1　今後の課題

今後，SmartTerminal Framework の上で動作する各種モジュールを増やし，SmartTerminal アプリケーションとして提供可能なサービスを増やしていく．SmartTerminal Framework をより広く利用可能にするには実装のしやすさなどを評価し，改良を加える必要がある．また，SmartTerminal の利用を促進するためには，今後以下の課題に取り組む必要がある．

パッケージの拡張

本実装では，クライアント端末間の通信，情報や機器機能，複数のサービスが利用する基本機能，および位置情報を必要最低限の機能として提供している．しかし，サービスが増えるにつれ，これらのパッケージ構成だけでは機能が不足する可能性がある．例えば，ユーザがおかれている状況を認識し，その状況を考慮してサービスを提供したい場合がある．そのためには，状況認識モジュールを実装し，一致する機能のパッケージに追加するのだが，パッケージが存在しない場合は新たに作成する必要がある．このように，サービスの必要に応じて，SmartTerminal Framework を拡張する．状況の認識以外には，個人のプレファレンスの利用などの機能の拡張も今後の課題の一つである．

ネットワークインフラの整備

SmartTerminal を実際に実用化するためには，インターネット接続を可能とするネットワークインフラおよび無線通信技術が必要不可欠である．複数の企業では，既にホットスポットや情報端末を提供している．今後このようなサービスが増えると共に，サービス提供者間での協力やサービスの相互運用が重要になっていく．例えば，ISP 間での提携により，ホットスポットからの ISP ローミングが可能になっている．しかし，ホットスポットの設置に関しては問題が発生している．

代表的な例として，先行研究で取り上げた MIS があげられる．MIS は，公共な場所で移動中に自由に高速にインターネットに接続できる環境づくりを目的としている．そのためにネットワークインフラとして無線基地局の設置を行っていた．その一環として駅に無線基地局を設置しようとしたが，下記のような理由により JR[8] に断られた．

- 無線のチャンネルは個数が限られているため，容易に他社に利用させるわけには行

かない

- JR が提供しているサービスと干渉し，障害が起きるかも知れない
- わざわざ駅に設置する必要がない

今後，SmartTerminal が普及していくためには，サービス提供者間の協力とネットワークインフラの整備が必要となっている．

ビジネスモデルの考慮

SmartTerminal を街中に設置した場合，サービス端末上で動作するサービスは複数のサービス提供者によって実装および運営されることとなる．そのためには，SmartTerminal 自身の設置，保守および運営を考慮したビジネスモデルも視野にいれる必要がある．

9.2 本論文のまとめ

携帯電話やホットスポットサービスの普及にともない，公共空間におけるユーザの計算機利用率が増加した．しかし，現在提供されている環境では利用可能なサービスは乏しく，機器が持つ入出力機能の制約が大きい．そのため本論文では，通信，情報および機器機能を提供する情報端末 SmartTerminal を提案し，SmartTerminal を構築するために必要となる基盤技術 SmartTerminal Framework の設計および実装について述べた．SmartTerminal ではサービスを実装するためのインタフェースや，管理するための機構を提供する．サービス提供者はこれらを利用して SmartTerminal のためのアプリケーションの開発を行う．

SmartTerminal Framework の利用方法を示すために，SmartTerminal アプリケーションとして SmartTaxi と SmartBus アプリケーションを示した．評価においては SmartBus アプリケーションを利用し，定量的に評価を行い，オーバーヘッドや SmartTerminal Framework の有用性を検証した．

謝辞

本研究を進めるにあたり，丁寧な御指導と数々の御助言を賜りました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，本論文の副査として御指導と御助言を頂いた，慶應義塾大学環境情報学部教授清木康博士，慶應義塾大学政策・メディア研究科特別研究教員助教授高汐一紀博士に深く感謝致します．

慶應義塾大学政策・メディア研究科博士課程の中澤仁氏，同博士課程の由良淳一氏，岩井将行氏には本論文執筆に至るまでの 4 年間，日頃から絶えざる御助言と励ましを賜りました．KMSF 研究グループ，徳田研究会の方々には研究生活の中で数々の刺激と知識を頂きました．公私共に励まし合い研究生活を送った慶應義塾大学政策・メディア研究科修士課程三川荘子氏，同修士課程鈴木雅子氏，並びに研究生活を共に過ごした同期の方々に深く感謝致します．本論文執筆にあたり多忙な中遠方より励ましを頂いた慶應義塾大学政策・メディア研究科博士課程今泉英明氏，本論文に専念できるようにサポートして頂いた同政策・メディア研究科修士課程古市悠氏，常に独特なおもむきで励ましを頂いた同環境情報学部高橋元氏，および徳田・村井・楠本・中村・南合同研究会のみなさまに深く感謝致します．

最後に，日頃から暖かく見守って頂いた両親，友人に感謝の意を表し，謝辞と致します．

平成 15 年 2 月 22 日
慶應義塾大学 政策・メディア研究科 修士 2 年
石井かおり

付 録A

A.1 IrdaConnectionクラスサンプルコード

```
package jp.ac.keio.sfc.ht.ksmf.smartterminal.connection;

import java.io.*;
import java.util.*;
import javax.comm.*;

public class IrdaConnection extends Connection
                            implements SerialPortEventListener {

  public static final String name = "IrdaConnection";
  public static final short ID = "1";
  public static final short MODULE_INACTIVE=0;
  public static final short MODULE_ACTIVE=1;
  public static final short MODULE_ERROR=2;

  public short status = 0;
  public boolean connection = false;

  CommPortIdentifier port;
  SerialPort serialPort;

  InputStream inputStream;
  OutputStream outputStream;

  ConnectionListener listener;

  public IrdaConnection(String comPort, ConnectionListener lis)  {
    init();
  }

  protected void init() {
    try {
      port = CommPortIdentifier.getPortIdentifier(comPort);
    } catch (NoSuchPortException npe) {
      npe.printStackTrace();
    }
    this.listener = lis;
  }

  protected void activate() {
    this.open();
    status = MODULE_ACTIVE;
  }
```

```
public void open() {
  try {
    serialPort = (SerialPort) port.open("irda", 2000);
  } catch (PortInUseException e) {
    String owner = port.getCurrentOwner();
    System.err.println("IrdaConnection: PortInUseException by " + owner);
    e.printStackTrace();
  }

  try {
    outputStream = serialPort.getOutputStream();
    inputStream = serialPort.getInputStream();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }

  try {
    serialPort.addEventListener(this);
  } catch (TooManyListenersException tme) {
    tme.printStackTrace();
  }

  serialPort.notifyOnDataAvailable(true);
  try {
    serialPort.setSerialPortParams(9600,
                                   SerialPort.DATABITS_8,
                                   SerialPort.STOPBITS_1,
                                   SerialPort.PARITY_NONE);
  } catch (UnsupportedCommOperationException e) {}

  this.start();
}

public void run() {
  try {
    Thread.sleep(5000);
  } catch (InterruptedException ie) {
    ie.printStackTrace();
  }

  connected = true;
  ConnectionEvent ce = new ConnectionEvent(this,
                       ConnectionEvent.CONNECTION_REQUESTED);
  listener.connectionRequested(ce);
}
```

```
public void serialEvent(SerialPortEvent event) {
  switch(event.getEventType()) {
  case SerialPortEvent.BI:
  case SerialPortEvent.OE:
  case SerialPortEvent.FE:
  case SerialPortEvent.PE:
  case SerialPortEvent.CD:
  case SerialPortEvent.CTS:
  case SerialPortEvent.DSR:
  case SerialPortEvent.RI:
  case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
    break;
  case SerialPortEvent.DATA_AVAILABLE:
    connected = true;
    String message = this.read();
    System.out.print(message);
    break;
  }

}

public String read() {
  byte[] readBuffer = new byte[20];
  try {
    while (inputStream.available() > 0) {
      int numBytes = inputStream.read(readBuffer);
    }
    return String.valueOf(readBuffer);
  } catch (IOException ioe) {
    return "IrdaConnection: No message";
  }
}

public void write(String message) {
  try {
    outputStream.write(message.getBytes());
    System.err.println("IrdaConnection: sent message " + message);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
}

public void close() {
  inputStream.close();
  outputStream.close();
  connection = false;
}
```

```
  protected void terminate() {
    serialPort.close();
    active = MODULE_INACTIVE;
    System.err.println("IrdaConnection: closing connection");
  }

  public boolean connected() {
    return connection;
  }

  protected short getStatus() {
    return status;
  }
}
```

参考文献

[1] 総務省情報通信政策局: 通信利用動向調査 (2001).

[2] エヌ・ティ・ティ・コミュニケーションズ株式会社: Hotspot. http://www.hotspot.ne.jp/.

[3] 西日本電信電話株式会社: フレッツ・スポット. http://www.ntt-west.co.jp/flets/spot/.

[4] Yahoo! BB: Yahoo! BB モバイル. http://www.bbtec.net/bbmobile/.

[5] インターネットコム株式会社, 株式会社インフォプラント: モバイル情報発信拠点としての「駅」の可能性. http://japan.internet.com/research/20020902/1.html.

[6] インターネットコム株式会社, 株式会社インフォプラント: ホットスポット、実際の利用はまだまだ? 利用経験は 10 % 未満. http://japan.internet.com/research/20020709/1.html.

[7] 総務省: 電子政府・電子自治体の推進のための行政手続オンライン化関係三法のポイント (2002). http://www.e-gov.go.jp/.

[8] 東日本旅客鉄道株式会社: 無線による、駅でのインターネット接続実験. http://www.jreast.co.jp/musenlan/.

[9] 電子情報技術産業協会: JEITA HOUSE. http://www.eclipse-jp.com/jeita/.

[10] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J. and Serra, B.: *people, places, things: web presence for the real world* (2000).

[11] Institute of Electrical and Electronics Engineers, Inc: IEEE. http://standards.ieee.org/.

[12] IEEE 802.11 Working Group: *IEEE 802.11*. http://standards.ieee.org/wireless/.

[13] Infrared Data Association: IrDA. http://www.irda.com/.

[14] Bluetooth Special Interest Group: *Bluetooth Specification* (2001). http://www.bluetooth.com/.

[15] Mobile Internet Services: genuine. http://www.miserv.net/.

[16] エヌ・ティ・ティ・ソルマーレ株式会社: Foobio. http://www.nttsolmare.com/top.html/.

[17] 日本コカ・コーラ株式会社, 株式会社エヌ・ティ・ティ・ドコモ, 伊藤忠商事株式会社: Cmode. http://www.cmode.jp/.

[18] 宇根正美, 尾田継之: 災害対応機能を有する街角情報端末等の開発, 県受託研究報告書 (2000).

[19] Raman, B., Agarwal, S., Chen, Y., Caesar, M., Cui, W., Johansson, P., Lai, K., Lavian, T., Machiraju, S., Mao, Z. M., Porter, G., Roscoe, T. and Mukund: The SAHARA Model for Service Composition Across Multiple Providers, *Proceedings of Pervasive Computing* (2002).

[20] Grimm, R., Davis, J., Lemar, E., MacBeth, A., Swanson, S., Gribble, S., Anderson, T., Bershad, B., Borriello, G. and Wetherall, D.: Programming for pervasive computing environments, Technical report, University of Washington, Department of Computer Science and Engineering (2001).