修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院情報システム学研究科　　情報システム基盤学専攻　　博士前期課程		
氏　　名	山本　峻丸	学籍番号	１３５３０３２
論 文 題 目	「気付き」を誘起する忘れ物防止支援システム		

要　旨

忘れ物をした経験は誰にでもある．日々必要な財布や携帯電話から，会議のための事前配布資料，雨の日における傘まで，忘れ物の種類は非常に多岐にわたる．「うっかり」忘れてしまうことを防ぐべく，記憶の補助をしてくれる秘書のような存在，すなわち外部脳が求められていると言える．

「忘れ物」という課題に着目した，パーソナルな持ち物管理システムはこれまでも数多く提案されてきた．多くが RFID システムを利用して持ち物の管理をしており，システム内の持ち物表と現在持っている物を比較し，外出しようとしているタイミングでシステムが警告を出すといったものである．しかし，これらは未だ一般に利用されるまでには至っていない．これは，2 つの手間が理由なのではないかと考えた．1 つめは，英数字の羅列である RFID と物の名称の対応表を作る手間．2 つめはどんな予定がある時にどんな物が必要であるかという表を作る手間である．「忘れ物」という，生起頻度の低いことに対してかける手間としては，2 つの表作成という作業はあまりに大きい．

そこで本研究では，「忘れ物があるか」「その忘れ物はどのような時に使うものであるか」という 2 点をユーザに提示すれば，忘れ物を効率的に防止出来るのではないか，という仮説を立てた．忘れ物が存在することをユーザに提示し，荷物の再確認を促すという機能だけに絞ることで，前述のような表作成を不要とする手法を提案する．この手法では，具体的な忘れ物の名称をユーザに提示することをしない代わりに，どのような時に必要な物であるかをユーザに提示し，忘れ物への気付きを誘起することに焦点を置いている．システムから「『○○』で使う物をお忘れではありませんか？」というようにユーザに問いかけることで，ユーザに持ち物の再確認を促すのである．

また，従来の提案では RFID リーダをゲート型にして，家の玄関などに設置することで持ち物を管理するものが大半であった．しかし，近年のスマートフォンでは NFC リーダが内蔵されているように，今後，アプリケーションの発展次第で他の周波数帯の RFID リーダもまた内蔵されることが期待できる．これを踏まえて，本研究ではユーザが RFID リーダを持ち歩く方式を提案した．

平成 26 年度 修士論文

「気付き」を誘起する
忘れ物防止支援システム

電気通信大学 大学院情報システム学研究科
情報システム基盤学専攻
1353032　山本 峻丸

主任指導教員　多田 好克　教授
指導教員　小宮 常康　准教授
指導教員　古賀 久志　准教授

提出日　　平成 27 年 1 月 26 日

目次

図目次

表目次

ソースコード目次

第1章 序論

1.1. 研究背景

ユビキタスコンピューティングという概念が登場して 20 年が過ぎた．この間に情報技術は飛躍的な発展を遂げ，あらゆる場所にコンピュータが埋め込まれるようになった．その存在を意識することなく，いつでも・どこでも・だれでもネットワークの恩恵が受けられるような世界が間近に迫っている．近い将来，これまでには想像もしなかったようなものにもコンピュータが組み込まれ，ネットワーキングの対象となることが予想される．

しかし，その実現にはハードウェア面とソフトウェア面双方の課題解決が欠かせない．ハードウェア面では小型化や電力確保の問題，低価格化といった課題があり，ソフトウェア面では多数の機器を接続するネットワーキングの技術やそれを活用するソフトウェアの開発が課題となる．特に後者，ソフトウェア面においてキラーアプリケーションの模索は重要と考えられる．キラーアプリケーションや，キラーサービスの登場は，ユビキタスコンピューティングそのものの発展を促すからである．

既に，一部で模索と実用化は進んでいる．国内外の小売店舗で使用されている無線タグを用いた物流管理がその一例である．しかし，そのほとんどがビジネスの現場を対象としていることを指摘しておかねばならない．これは，ビジネスの現場において業務改善や効率化は直接的に利益に繋がるため，確実に効果があるものであれば導入に対して積極的であるためだと考えられる．すなわち，家庭よりも金銭的なハードルが低くなりやすく，性能や安定性などへの要求が高くなりやすい．

ビジネスをターゲットしたアプリケーションを家庭にそのまま導入することは困難である．ビジネスの現場における需要と家庭における需要は一致するとは言えないためである． 例えば，先に論じたような無線タグの物流管理では，物の位置をリアルタイムで管理することに焦点が置かれているが，家庭においてこのような機能が必要かは疑問である．物を動かすのは家族に限られ，家族に聞けば大概は解決できる．家庭で求められるのは，例えば外出先から冷蔵庫に入っている食材が何であるかであったり，外出時に忘れ物がないかを確認できるかであったりというような機能である．物流管理システムでは，過大な機能に対してキャリブレーションの手間や過大な導入費用を払わなければならない．やはり，ビジネス向けとは異なる視点でもって，ユビキタスコンピューティングの恩恵を家庭で得られるアプリケーションを考えなければならない．

本研究では，特に需要が高いと考えられるパーソナルな物管理について，無線タグ

を用いてユーザが日々の持ち物を管理する際のサポートを行う手法を提案する．この手法は，無線タグで全ての機能を提供しようとするのではなく，あくまでユーザを中心において「気付き」を与えることに注力する．

本手法では，既存手法とは異なり，持ち物の名称をユーザに登録することを求めない．また，スマートフォンに内蔵されたスケジュールソフトからユーザの予定を取得し，必要な持ち物を自動的に推測する．これにより，既存の持ち物管理手法に比べ，導入コストやメンテナンスコストを大きく抑えられることが期待できる．

1.2. 技術的背景

本節では，ユビキタスアプリケーションなどで用いられている技術のうち，特に持ち物管理に関係性の深い個体識別について説明する．さらに，本研究で使用する Passive RFID についての技術的制約などについて調べる．加えて，本研究で通知に用いるウェアラブルデバイスについても調べる．

1.2.1. RFID システムによる個体識別技術

個体識別技術とは，生物や物体を識別するための技術である．例えば，近年食のトレーサビリティが注目されているが，牛の場合，子牛が産まれてすぐに個体識別番号が書かれたタグを耳に取り付けて識別することが義務付けられている[1]．これも個体識別技術の 1 つと言っていいだろう．他の例をあげると，卸売業の倉庫においては最低年に 1 回は棚卸しを行う必要があるが，専用の機械を使用し，何がいくつ存在するのか一気にカウントを行うことがある．これも個体識別技術の活躍する場の 1 つである．

後者の例で多く用いられているのが，一次元・二次元コード（各社製品例[2], [3], [4], [5]），そして無線通信を用いた非接触個体識別技術の一種，RFID 技術（各社製品例[6], [7], [8]）である．RFID 技術では，RF タグとリーダの 2 つの機器が用いられる．RF タグは自身を識別するコードなどを発信し，リーダがその信号を読み取る．

RF タグにはいくつかの種類があるが，大きく分けて Active 型と Passive 型の 2 種類が存在する．これは電源の取得方法による分類で，Active 型は電池を内蔵してそれにより電波を発信するのに対し，Passive 型では受信した電波を電力に変換して発信する．電磁誘導現象などを用いている性質上，Passive 型では電力源となる電波を発信するリーダから大きく離れると動作できない．それに対し，Active 型では内蔵電力をもとに動作するため，常に電波を発信しており，Passive 型に比べ長距離での通信も可能である．

日本国内において RFID カードの一種である FeliCa カード[9]による電子マネーが，磁気カードから急速にシフトしたように，あるいはニュージーランドやオーストラリア，カナダなどで牛の個体識別に RFID を使用することが義務付けられたように，RFID はこれまでの個体識別技術を飲み込むほどの力を持っている．その一方で，よりプライベートな場においては，未だ RFID 技術の恩恵を充分に受けられているとは言い難い． RFID 技術が家庭の場に普及するためには，既存技術では提供することのできなかった価値を提案することができるキラーアプリケーションが求められている．

1.2.2. Passive RFID 技術

先に議論したように，Passive RFID とは外部から電力供給を受けて動作するタイプの RFID のことである．従来，電力供給手法が電磁誘導（磁力による誘導起電力を用いて RF タグに電力を供給する）に限られていたことで通信距離が最大 1m 程度と非常に限られていたが，近年電波方式（リーダから無変調波を射出し，RF タグのアンテナで受け取り，整流回路で直流に変換する）が実用化されたことで，リーダの出力や環境次第では 5m を超えるような遠距離でも通信が可能となった．これらの特徴については，表 1.1 にまとめた．電磁誘導方式と電波方式は，仕組みとしては大きく変わらないが，表 1.1 で示したように，電波方式は電力の利用効率が非常に低い．このため，電波方式で受け取れる電力ではタグ内のチップを動作させるのに不十分だったためである．

電波によって電力を確保し，チップを動作させ，応答を返すという仕組みであるため，Passive RFID は使用する電波帯域によって大きく特性が異なる．表 1.2 に周波数帯別の特徴を纏めた．なお，表中の仕様は，各周波数帯を利用する規格のうち特に多く使われているものについて記載している．

このなかでも，近年注目されているのは UHF 帯を使用するものである．これは，タグのサイズや価格，通信距離のバランスが最も優れているためである．

技術的仕様とは別の視点として，日本国内における電波法上の規制がある．例えば UHF 帯は構内無線局（最大出力 1W）もしくは簡易無線局（同 250mW），特定小電力無線局（同 250mW）として利用することができるが，構内無線局では屋外では利用できず，また，構内無線局や簡易無線局では個別に免許が必要である．このことから，1.1 で議論したようなパーソナルな物管理システムを提案するにあたっては，特定小電力無線局の範囲で利用できることが望ましいといえる．

表 1.1　RF タグの電力供給の方式．（東芝の資料[10]を元に作成）

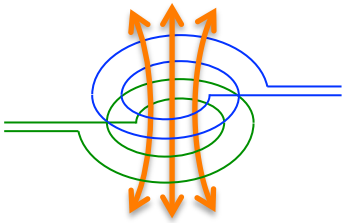
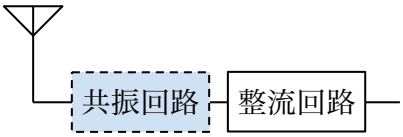
方式	電磁誘導	電波
仕組み		
電力を送信できる距離	数 mm～10cm	数 10cm～数 m
周波数	長波	中波～マイクロ波
電力の利用効率	70～90% （残りは主に熱になる）	1%以下 （残りは電波になる）

表 1.2　使用する周波数帯によって異なる PassiveRFID の特性．

周波数帯	134.2kHz	13.56MHz	900MHz（UHF）	2.45GHz
通信距離	～50cm	～1m	～5m	～3m
通信速度	4kbps	26kbps	40kbps	40kbps
動作方式	電磁誘導	電磁誘導	電波	電波
備考	初期からある方式だが，小型化が困難なことから近年ではあまり利用されていない	13.56MHz 帯には近接型カード規格（NFC 等）もあり，これは通信距離 20cm 以下，速度 106kbps である	国内では 2005 年電波法改正により解禁	400 μm 四方の製品が販売されているなど，超小型化が可能

1.2.3.　ウェアラブルデバイス

ここ 1,2 年ほど，ウェアラブルデバイスと呼ばれる分野がにわかに盛り上がっている．同分野はこれまで非常に製品化例が少なく，一般にウェアラブルデバイスを手にすることは困難であったが，国内外で複数製品の一般販売が始まり，手に取りやすい状況ができあがった．

ウェアラブルデバイスとは，広義に腕時計やメガネ，衣類のように身につけることができるデバイスのことを指す．ウェアラブルデバイスはウェアラブルコンピュータとも呼ばれ，ユビキタスコンピューティングの一部と見なされており[11]，ウェアラブルデバイス市場の隆盛は即ちユビキタス社会の到来を示すものとも言える．

既に数多くの製品がリリースされており，メガネ型と腕時計型が特に製品が多い．メガネ型は HMD（ヘッドマウントディスプレイ）と呼ばれるジャンルの製品が多く，Oculus VR 社の Oculus Rift[12]や Google 社の Google Glass[13]，セイコーエプソン社の MOVIRIO シリーズ[14]が有名である．腕時計型としては，Google 社のモバイル端末用 OS である Android 向けの Android Wear 端末が数多く発売されている[15],[16]他，Aliph 社の Jawbone UP シリーズ[17]などの活動量計製品が目立っている．また，Apple 社は自社のスマートフォン向けとして Apple Watch[18]を 2015 年に発売することを発表している．この腕時計型デバイスを，特にスマートウォッチと呼ぶ．

ここ 2 年ほどで発表されたデバイスは非常に多く，ここで挙げたのはごく一部に過ぎない．まだ市場は立ち上がったばかりであることを考慮すると，今後さらに製品が増え，日常生活の中で見かけることも増えることが期待できる．

これらのデバイスは，その多くが，単体で動作するというよりもスマートフォンといった外部機器と連携して動作するものが多い．小さいデバイスが多いため，単体で使用するには操作性が悪かったり，提示することができる情報量が少なすぎたりすることがこの理由であると考えられる．本研究においても，スマートフォンと連携させた Android Wear 端末を利用する．

1.3. 研究目的

本研究は，ユーザの「気付き」を誘起して忘れ物の防止を支援するシステムを構築するものである．提案手法では事前に持ち物のデータベースを作成したり RFID と物名称の対応表を作成したりする必要がなく，利用開始時の時間的コストが飛躍的に削減できることが期待できる．

詳細は後ほど 2.6 節で比較するが，これまで提案されてきたシステムでは，持ち歩く物に対して予め RF タグを貼付し，その RF タグの ID と物名称の対応表を作成していた．この対応表を利用して，日々必要な物（＝RFID）をシステムに登録しておき，玄関などに設置されたゲート型 RFID リーダで持ち物とその日必要な物を比較し，足りないものがあればアラートを出すといった形態である．

しかし，「忘れ物」という，生起頻度の低いことに対してかける手間としては，物名称と RFID の対応表及び日々の持ち物表作成という作業はあまりに大きい．得られるものとかけるコストのアンバランスさは，ユーザにとって使い勝手がいいとは言い難

い．

そこで本研究では，忘れ物が存在することをユーザに提示し，ユーザに荷物の再確認を促すという機能に絞ることで，前述のようなリスト作成を不要とする手法を提案する．この手法では，具体的な忘れ物の名称をユーザに提示することをしない代わりに，どのような時に必要な物であるかをユーザに提示し，忘れ物に気付かせることに焦点を置いている．システムから「『◯◯』で使う物をお忘れではありませんか？」というようにユーザに問いかけることで，ユーザに荷物の再確認を促すのである．

また，従来の提案では RFID リーダをゲート型にして，家の玄関などに設置することで持ち物を管理するものが大半であった．しかし，近年のスマートフォンでは NFC リーダが内蔵されているように，今後，アプリケーションの発展次第で他の周波数帯の RFID リーダもまた内蔵されることが期待できる．これを踏まえて，本研究ではユーザがリーダを持ち歩く方式を提案する．

1.4. 研究課題

本研究では，日々の持ちだされた RF タグとその日あったことをログとして保持し，そのログを利用して，どんな条件下で物が持ち出されるか推定することに取り組む．併せて，モバイル RFID リーダを用いて持ち物を管理することが可能であるかに関しても議論する．

1.5. 本論文の構成

本論文の構成は以下の通りである．

第 1 章　序論
第 2 章　「気付き」を誘起する忘れ物防止支援システム
第 3 章　システム設計
第 4 章　システムの実装
第 5 章　評価実験
第 6 章　結論

まず第 1 章では，本研究の技術的な背景について議論し，さらに目的や課題について論じる．続いて第 2 章で本研究のテーマである忘れ物について分析を加え，想定する環境やシステム構成について論じた後，関連する研究について調べる．第 3 章では，RFID システムから得られるデータを如何に利用し，どのような条件下でどの物が持ち出されるかを如何に推定するか，その手法について議論する．これを踏まえ，第 4 章でより細かく実装面の議論を行い，第 5 章で提案手法を評価する．最後に第 6 章で

……

本研究の成果をまとめ，今後の展望について議論する．

第2章 「気付き」を誘起する忘れ物防止支援システム

本章では，まず忘れ物についての分析を加えた上で，本研究の「気付き」とはどのようなものであるかについて議論する．さらに，提案する「気付き」を誘起する忘れ物防止支援システムがどのような環境下での使用を想定するか論究し，最後に関連研究を分析する．

2.1. 持ち物についての分析

忘れ物について分析をする前に，持ち物というものについて分析を行いたい．

人は様々な物を持ち運びながら行動する．財布や携帯電話，化粧用品に会議資料，ポストに投函しなければいけない葉書，スーパーに行くときの買い物メモ，そして買ってきた食材と，一括りにするのが躊躇われるほどに様々である．これらの持ち物は，様々な目的や理由があって持ち歩かれるものだ．なにかの予定で必要であったり，雨が降るから必要であったり，スギ花粉が飛散しているから必要であったり，料理をするから必要であったりと，それぞれの物がそれぞれの理由・目的のために持ちだされている．この，持ち物に影響を与える理由・目的のことを，本研究では「事象」と呼ぶ．

当然のことだが，事象と物は 1 対 1 ではない．図 2.1 は，事象と持ち物の対応を示す例である．左に書かれた事象 A から右に書かれた物 a に矢印が伸びていれば，その物 a が事象 A で必要であることを示す．この例で言えば，事象「雨予報」と持ち物「傘」は 1 対 1 の関係であるが，事象「会議」では持ち物「ノート PC」「変換アダプタ」「事前配布資料」の 3 つが必要だ．その一方で，持ち物「ノート PC」には，事象「会議」「出張」「ゼミ」から矢印が伸びている．先に議論したように，持ち物は事象によって決定されるのであるから，その日の事象が決定すれば必要な持ち物を決定することができるが，持ち物のリストを元に事象のリストを復元することはできず，また，どの物がどの事象で必要であったのかを単純に推測することはできない (図 2.2)．本研究では，この対応を事象と物の一方向的対応性と呼ぶ．

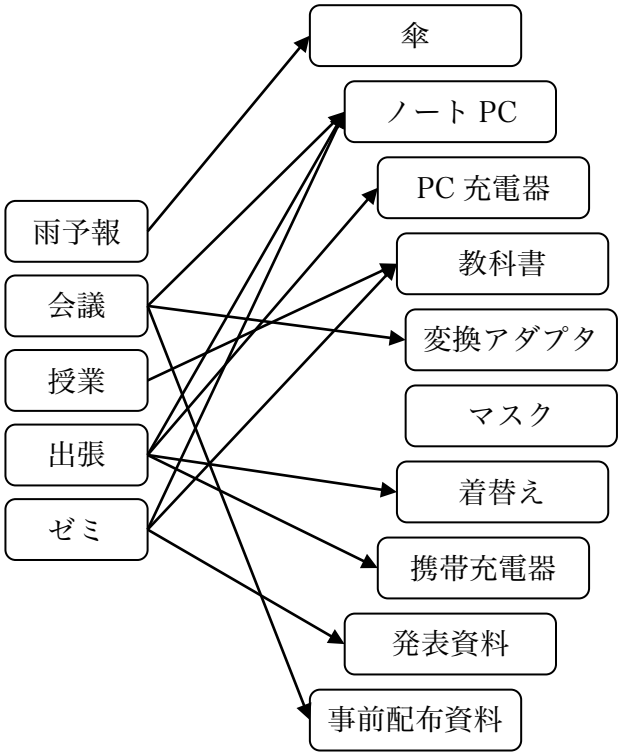
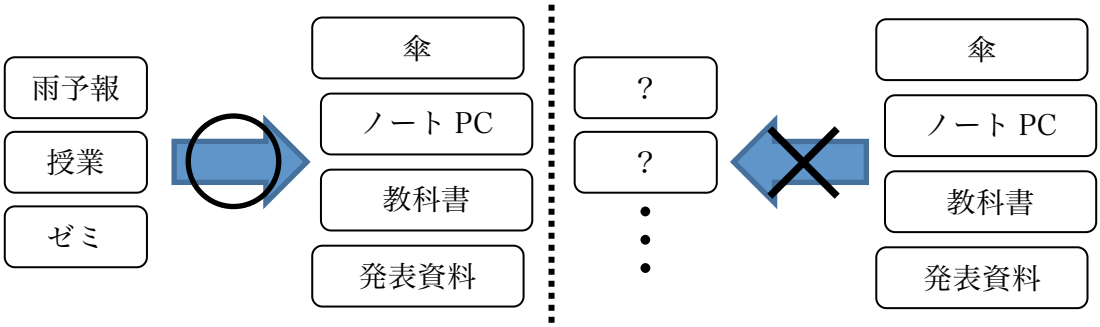


図 2.1　事象と持ち物の対応例.



事象のリストから持ち物リストは生成できる（左）が,
持ち物リストから事象のリストを生成することは一般にはできない（右）

図 2.2　事象と物の一方向的対応.

2.2.　忘れ物についての分析

その日必要な持ち物を家に忘れてきてしまい，冷や汗をかくような忘れ物をした経験は，誰にでもあるだろう．日々持ち歩く必要のある財布や携帯，定期入れといったものから，会議のための事前配布資料，雨の日に欠かせない傘まで，忘れ物の種類は

非常に多岐にわたる．

一般に「忘れ物」というと，上記のような必要な物を持ち出し忘れたというタイプの忘れ物と，電車内などに持ち物を置いてきてしまうといった忘れ物の両方が想像される．

まず，必要な物を持ち出し忘れたというタイプの忘れ物は，大きくわけて２つの原因が考えられる．まず１つめは，その日にある事象の存在自体を忘れている，もしくは知らないということである．例えば,「今日会議があるのを忘れていて，会議で必要な事前配布資料を忘れた」という忘れ物や,「雨が降ってきたのに，家から傘を持って来なかった」という忘れ物がこれに相当する．

２つめが，事象自体の存在は覚えているものの，そこで必要な物を持ってこなかったというタイプの忘れ物である．例えば,「出張に出かけるのに PC の充電器を忘れてきた」「A プロジェクトの会議資料を持ってきたつもりで B プロジェクトの資料を持ってきてしまった」というような忘れ物がこれに相当する．

子どもの頃であれば，母親がこのような忘れ物を指摘してくれたりするものであった．「今日月曜日だから体育あるのでしょ？忘れ物ない？」といったような形である．社会人であっても，自分の「うっかり」を防ぐべく，持ち物の再確認を促してもらえる環境があれば，忘れ物の低減に繋がるのではないだろうか．記憶の補助をしてくれる秘書のような存在，もしくは外部脳とも言えるものが求められていると言える．本研究で特に注目しているのが，このタイプの忘れ物である．

一方，もう 1 つの「忘れ物」，すなわち置き忘れについても，ユーザへの再確認を促す問いかけは有効であると考えられる．例えば,「財布を席に置いたまま退社してしまった」などといった状況は，持ち物を再確認すれば気付けるのではないかと推測できる．本研究ではこちらの忘れ物について直接的には取り組まないものの，将来的な拡張によって対応が可能な設計を行っている．

2.3. 個人の物管理における「気付き」

2.1 節で行った分析から，本研究では 1 つの仮説を立てた．忘れ物を防止するに際し，具体的に何を忘れているかユーザに１つ１つ説明を行わなくても，注意喚起行えば,充分に忘れ物を防止するための外部脳足りえるのではないか,というものである．具体的には，物の名称をユーザに与えることをしなくても,「忘れ物があるか」「その忘れ物はどの事象で使う物か」という２点を人に提示できれば，効率的に再確認を促すことができ，忘れ物に「気付く」ことができるのではないか，と考えた．

この仮説に沿い，また仮説を確認する意味で提案するのが,「『気付き』を誘起する忘れ物防止支援システム」である．ここで言う「気付き」とは，ユーザも知らない事

象と持ち物の関係性を解き明かし，ユーザに対して提示するという意味合いではない．あくまでシステムの中心にユーザをおき，ユーザにとって既知であることをついうっかり忘れていた際に，それに対する注意を喚起するという意味での「気付き」である．

2.4. 「気付き」を誘起する忘れ物防止支援システムの想定環境

本節では，本研究の前提として想定している環境について議論する．持ち物を管理するにあたって，固有の ID を持つ RF タグを用いる．本研究では，持ち歩く可能性のある物 1 つ 1 つにこの RF タグを貼付することで持ち物を管理する．なお，RF タグでも，特に UHF 帯（920MHz）を利用する RF タグを想定している．この周波数帯を利用する RF タグは， 2.45GHz 帯を使った RF タグなどに比べ，アンテナが大きくなりがちではあるが，シールとして書類等への貼り付けも可能な製品も販売されている（例えば，富士通フロンテック社の「書類管理用ラベルタグ[19]」（図 2.3）は，厚みがほとんどなく，ロール形式のシールとして提供されている）．これらの製品を使えば，物を持ち歩く際にタグが邪魔となるような事態の考慮は不要である．

この RF タグの電波を日常的にセンシングするため，ユーザはモバイルリーダとスマートフォンを持ち歩く．据え置き型のリーダは必要としない．モバイル型の RFID リーダは既に様々なメーカから製品化されており，ユーザの環境に合わせて選択することが可能である．一緒に持ち歩くスマートフォンは，RFID リーダと接続して持ち物の管理を担う．

今回は RFID リーダとスマートフォンをそれぞれ持ち歩くこととしたが，物流現場向けとして，以前より Windows CE などを搭載したハンディターミナル型 RFID リーダが数多く製品化されている．2010 年には日立製作所と KDDI により UHF 帯 RFID を読み込むことができる携帯電話[20]（図 2.4）が発表されている他，スマートフォンのイヤホンジャックに刺して利用できる RFID リーダ[21]も販売されている．現時点では一般向けに販売されているスマートフォンに UHF 帯 RFID リーダは内蔵されていないが，今後のアプリケーションの発展・普及次第で内蔵されることが期待できる．

以上に加え，ユーザへのアラートを出すためのデバイスとして，スマートウォッチをユーザに使ってもらう．今回想定するような端末は，1.2.3 節で調べたように，ここ 1〜2 年で非常に手に入りやすくなった．通知を出すだけならばスマートフォンのみでも出すことができるが，ユーザが画面を見ていなかったり，バッグ内にしまっていてバイブレートに気付かなかったりしてしまう可能性が否定できない．その点，スマートウォッチは身につけているものであるから，バイブレートすれば気づいて画面

を見てもらえるだろう．メガネ型デバイスについても同様の効果は期待できるが，現在の製品の多くがよりリッチなコンテンツを表示することに重きをおいており，常時身につけたりするには不向きと言わざるを得ない．

以上を合わせると，ユーザが持つべきものは RFID リーダとスマートフォン，スマートウォッチに加え，RF タグが貼り付けられた持ち物が入ったバッグである．この利用イメージを図 2.5 に示す．

図 2.3　富士通フロンテック社の販売する書類管理用ラベルタグ．
（出典：富士通フロンテック プレスリリース[22]）

図 2.4　RF タグの読み取りに対応した携帯電話．（出典：日本経済新聞 [20]）

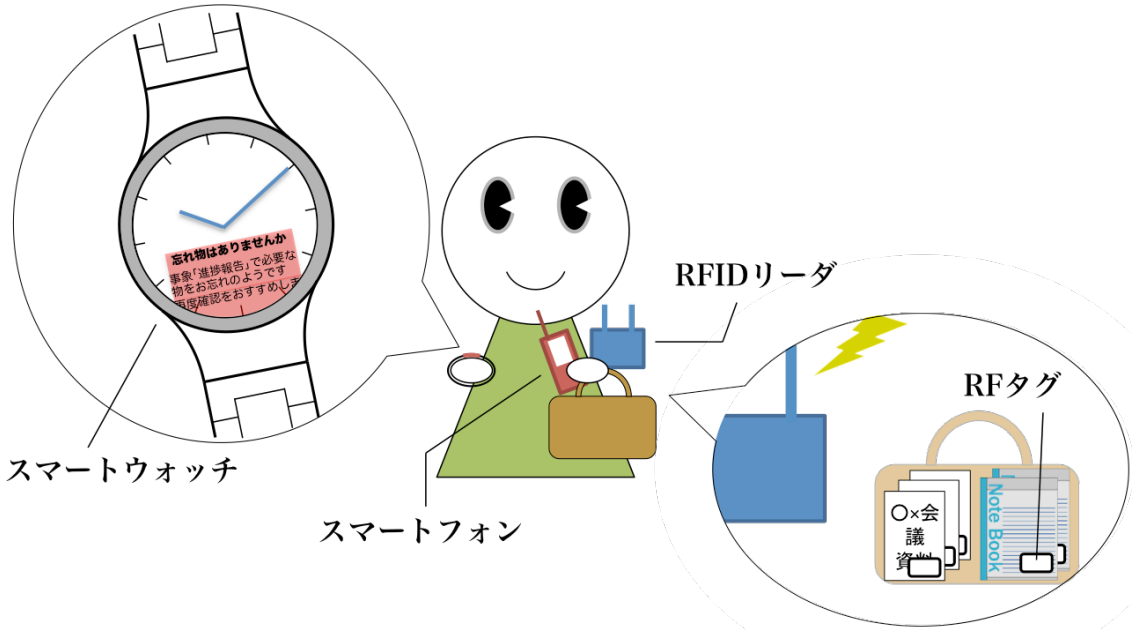


図 2.5　提案システムの利用イメージ.

2.5.　「気付き」を誘起する忘れ物防止支援システムの想定システム構成

本システムは，大きく分けて 3 つの部分から構成されている.

- 現在の周辺状況（持ち物など）を把握する部分
- 事象と持ち物の関係性を推測する部分
- 今後の事象を取得して必要物を推測，忘れ物をユーザに通知する部分

1 つめは，モバイル RFID リーダを用いて近隣 RF タグと通信を行うものである．まず受信したデータを補正（穴埋め）し，3 つめの部分に渡すと同時にデータベースへの保存を行う．本システムは，2.4 節でも議論したように UHF 帯 RFID システムを使用することを想定するものである．これは，1.2 節で調べたように，タグのサイズや価格，通信距離のバランスが優れていることから本システムにより適した手段として選択したものであり，必ずしも RFID を利用することに限定されたものではない．今後の技術動向により，他の個体識別技術を使用することも可能であると考えられる．その際は，この現在の周辺状況を把握する部分を，選択した個体識別技術に合わせることとなる.

2 つめは本研究の鍵となる部分である．2.3 節で議論したように，本研究は物の名

称を使わずに，どのような事象があるときに持ちだされている物であるかをユーザに対して提示する．これは，確率の考え方を用い，過去の記録から，どのような事象があるときに必要としている物であるかを推測する．

3 つめは，2 つめから得られた情報を元に，本日の事象リストと組み合わせることで必要であると考えられる物を推測，持ちだされていなかった場合にアラートを出す部分である．今回，先に調べた腕時計型ウェアラブルデバイスを用いることで，ユーザに対してより気づきやすい通知を行うことが可能になる．

これらをもう少し細分化すると，図 2.6 のようになる．この図からわかるように，処理のフローはリアルタイムに行うものと 1 日 1 回行えば良いものがある．先の 3 分類で言うならば，1 つめ・3 つめがリアルタイムで行うべきことであり，2 つめが 1 日 1 回行えば良いことである．青枠内はいずれもユーザの持ち歩くスマートフォン内で処理が行われ，サーバなどは必要がない．

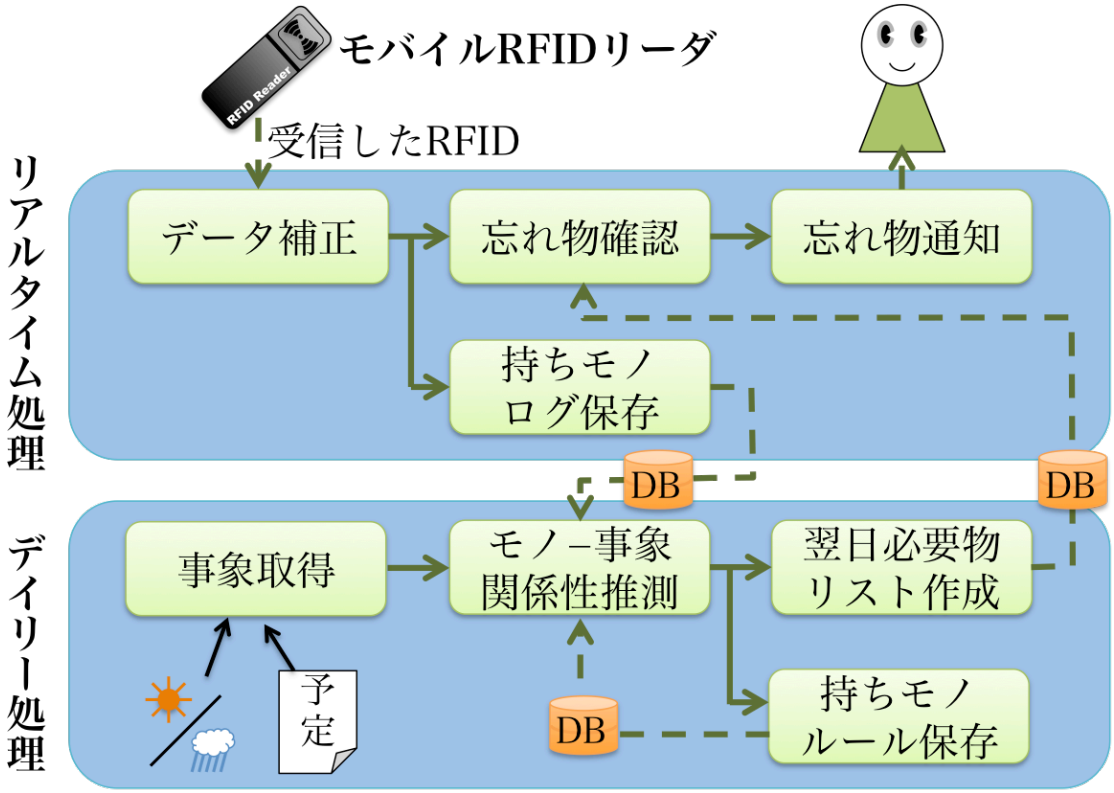


図 2.6　提案システムの概要図.

2.6. 関連研究・事例

物探しや物品管理，忘れ物の防止を目的とした研究や製品は数多く存在する．本節ではそれらについて説明する．

2.6.1. 物探し支援システム

忘れ物と同じように，探し物が見つからないという経験も，誰にもあるものだろう．物の位置座標を計測するには，RF タグの電界強度を使って RSS（Received Signal Strength，距離の二乗に反比例して電波が弱くなることを用い，受信電波強度から距離を推定，三辺測量する）や TOA（Time Of Arrival，距離に比例して電波の滞空時間が延びることを用い，滞空時間から距離を推定，三辺測量する），TDOA（Time Difference Of Arrival，リーダが電波を受信した時間の差から距離を推定，三辺測量する）や AOA（Angle of Arrival，リーダに到達する電波の角度を計測し，三角測量する）で測位する手法が一般的である．この他，タグから送信した超音波・赤外線を検出して位置を推定する手法[23],[24]，またそれらとカメラ画像からの動画像処理を組み合わせて位置を推定する手法[25]，モバイル RFID リーダを持ち歩いてタグの出現パターンからタグの相対的位置を推定する手法[26]など，様々な手法が提案されている．

山本ら[27]は，あらゆるオブジェクトに加速度センサと位置センサを取り付け，3 次元位置センサと超指向性スピーカを組合せたシステムで，放射される超音波を床に当て，音の道筋を作ることでユーザを目的物へ誘導する手法を提案している．この研究はユーザに通知する手法に焦点をおいたもので，センサ類から得られる物の絶対座標と，通常人間が利用している相対的な座標との間を埋めようとしているものだと言える．

同様の着目を行った研究としては，中田ら[28]による，探し物のある付近にスポットライトを照射する手法の提案がある．この研究では，物に Active 型 RF タグを取り付けておき，測定した位置がユーザにわかりやすくするためにスポットライトを用いた．音と光という差はあるが，前者と近い発想だと言えよう．

一方，測位手法の方に，従来と異なる発想を取り入れた研究も多くある．

西原ら[29]は，物を移動させる存在は人間に他ならないことに注目した．物に振動センサを内蔵した Active 型 RF タグを貼り付け，移動中の物を認識，人間の発する遠赤外線から人間の位置（＝物の位置）を検出して，動いている物の ID とその位置を推定する手法を提案している．

佐竹ら[30]は，日常生活での物探し支援システムを提案している．探しものが過去に置かれていた場所を室内に設置されたカメラの画像内にマークし，検索を行う．これ

は言わば，座標を画像と結びつけることで，人にとってわかりやすい相対的な座標に変換していることに等しい．

小松崎ら[31]は，複数の収納箱に物が収納されていることを前提に，BoxFinder という物探し支援システムを提案している．デジタルカメラで 2 次元コードを付けた収納箱の中身を撮影することで，システムが 2 次元コードを認識し，箱番号と写真を関連づけて保存する．これにより，ユーザは手軽に箱内の画像を閲覧できる．さらに，手動で箱の位置を登録することもできる．また，中川ら[32]は，BoxFinder で 2 次元コードではなく RF タグを用いる手法を提案した．これら収納箱による管理も，システムの持つ絶対座標と写真という相対環境を結びつけているものだと言えよう．

ただ検索性を高めるのではなく，過去の自分自身の行動を顧みることで物を発見できるはずという考えで提案されている研究もある．上岡ら[33]は，日常生活で用いることを前提としたウェアラブルシステムを提案している．ユーザが常にカメラを装着し，身の回りを録画することによって，探しているシーンをユーザに確認させ，その時点の体験を思い出す活動を支援することによって物探しを支援する．

このように探し物という分野においても数多くの研究がなされており，物の保管をシステムが補助することは，大きなニーズのあることだと言えよう．しかし，物の位置をシステムが管理するには，未だ大きな金銭的・時間的コストの問題がある．一般家庭で広く使われるようになるためには，その点の解決が欠かせないだろう．

2.6.2.　物品管理システム

本研究はプライベートな持ち物を対象とし，また物の保管場所などを管理するものではない．しかし，倉庫などで使われている物品管理システムも，物を管理するという意味で類似した部分が存在する．本節ではこれらについて議論する．

倉庫などで使われているシステムは，多くが RF タグを用いている．これは．複数個同時に読み込むことが可能であり，かつ見えていない場所にあっても読み込むことができるという RF タグの特性が活きやすいためである．

NEC は SmartAsset[34]という物品管理システムを提案している．これは UHF 帯 RF タグとリーダ内蔵 PDA を使用し，棚卸を容易にする．オプションとして物品の貸出管理や持出管理も可能になっているほか，リーダ内蔵 PDA で探したい物品を選択し，ダンボールなどにかざすと箱のなかに入っているか調べる機能もある．1 次元/2 次元バーコードを用いることも可能である．

代官山　蔦屋書店では，UHF 帯 RF タグ約 80 万枚を用いて在庫管理を行なっている[35]．図 2.7 のように RF タグを内蔵したシールを商品に貼り付け，RFID リーダ内蔵の棚を使い，リアルタイムに所在地を把握する．これにより客が求める商品の正確な位置を提示することが可能になった他，スマートゲートと組み合わせたことで盗難

を防止することにも繋がった．同時に，リーダを内蔵したセルフレジを多数設置し，レジ操作を効率化した．

日立製作所は AirLocation[36]を提案している．これは 5 台の無線 LAN アクセスポイントを空間内に設置して無線 Lan 端末の場所を測位する．測位精度は 1〜3m 程度と，専用品を使うものに比べて少し劣るが，一般的に良く用いられる機材を用いることで，専用機材を用いるものに対してコストを削減した．

このように，工場や倉庫，店舗といったビジネス寄りのソリューションは，既に研究の域を出て製品化の領域に入っている．しかし，いずれもパッケージとしてすぐ利用できるような形にはなっておらず，導入時には念入りなキャリブレーションが欠かせない．また，わずかに棚を動かした程度であっても，電波環境は大きく変化しうるので再度のキャリブレーションが必要である．このため，金銭的・時間的コスト共に．一般の家庭などで導入は到底不可能である．

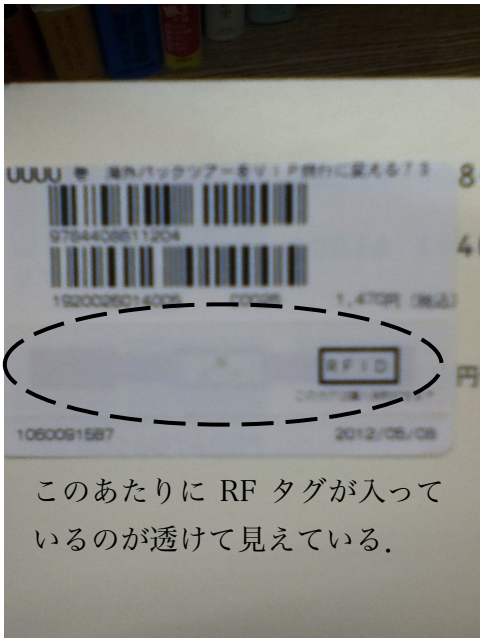


図 2.7：蔦屋書店にて商品に貼り付けられた RF タグ．

2.6.3.　忘れ物に注目したシステム

本研究と同じく，忘れ物に注目した研究もこれまでいくつかなされてきた．

Mik Lamming ら[37]の Forgot-me-not は，現在でいうところのウェアラブルデバイスを用いたライフログの考え方を提案するものである．腕時計型のデバイスにアイコンを用いたインタフェイスを表示し，過去の予定や人からその時に使った物を絞り込

むというものである．しかし，この論文はデータの収集については将来的な技術発展に託した形であり，主にインタフェイスを提案するに留まっている．

Bradley J. Rhodes ら[38]の The Remembrance Agent は，テキストとして行動や物理センサの情報をログに残し，単語の出現頻度（Term Frequency）と逆文書頻度（Inverse Document Frequency）を用いて現在の状況に最もマッチする情報をヘッドアップディスプレイに表示する研究である．このシステムでは，Active Badge というものを使って持ち物を管理する．Active Badge は初期のユビキタスコンピューティング分野における有名な研究であり，5cm 角程度の Badge と各部屋に取り付けられたセンサと赤外線通信を行い，どこに Badge が存在するかをシステムに認識させるものである[39]．Forgot-me-not に比べ，自動で物が認識できるようになった点は注目すべきである．しかし，先に論じたとおり，Active Badge は建屋の側にセットアップが必要である他，Badge が大きい，バック内などでは検知ができないといった問題があるために，持ち歩く物 1 つ 1 つで使用するには充分な技術とは言えない．また，持ち物のデータベースを作るという手間がかかってしまうほか，様々なデータを使ってそれらの共起確率を求めるというものであるために偽陽性を引き起こしやすいという問題がある．

The Remembrance Agent における Active Badge 由来の問題を解決した研究として，Gaetano Borriello ら[40]の Reminding Engine がある．この研究では Passive 型 RFID を使用しており，家の玄関にゲート型リーダを設置することで，持ち物を管理する手法を提案している．どんな物をいつ，どこからどこへ持っていかねばならないかという情報を専用言語で記述することでその日に必要な持ち物を割り出し，RFID システムによって判別した持ち物と比較することで，忘れ物に警告を出すというものである．しかし，専用言語によるルールを記述するのは大きな手間となるだろう．また，家庭の玄関にゲートを設置することも，費用面などで導入障壁となりうる．

Hui-Huang Hsua ら[41]は，曜日をベースに持ち物を推定することで予定と持ち物を関連付け，忘れ物を防止するシステムの提案を行った．どのような予定でどのような持ち物が必要であるか，システムにユーザが教える手間を省くための提案であるが，予定は曜日とだけ結びつくものではない．筆者らの想定するユーザは学校に通う学生であるためこのような手法でも一定の効果が見込まれるが，一般に使ってもらええるシステムとするには，また異なるアプローチを組み合わせる必要があるだろう．

照屋のぞみら[42]は小学生を想定ユーザにした．先生が必要な持ち物をシステムに入力すると，児童のランドセルに取り付けられたディスプレイにその通知がなされ，NFC タグによって判定された現在の持ち物と比較して忘れ物を防止するシステムの提案である．しかし，これは言うまでもなく管理する側（先生）と管理される側（児童）という関連性があるために使用可能な手法であり，一般に対して適用できる手法

ではない．

このほか，浜野悠介ら[43]による重量計を使ってユーザがどんな物を持っているかを推測する手法の提案や，Tiancheng Zhang ら[44]による RF タグを貼付された物体の包括関係（ノート PC に対する充電器といったような，物の親子関係）を推測するための SPIRE 演算の提案といった，人の忘れ物に注目した要素技術の開発が進められてきた．

様々に研究がなされてきていることから，忘れ物に対する潜在的な需要があることは充分に示されているといっていいだろう．しかし，未だ我々の生活にその恩恵は受けられていない．この原因として，大きく言うと 2 つの「手間」が問題であると私は考えている．

まず 1 つめは，「RFID と持ち物の対応表を作る手間」である．人が持ち歩く物は様々だ．小物を含めれば，数十個の物を日々持ち歩いている人もいるだろう．しかも，それはずっと固定されているものではなく，長い時間の中で変化していくものだろう．新しい物を持ち歩くようになったからデータベースに登録を，というのはスマートであるとは言い難い．解消されて欲しい手間だろう．

もう 1 つは「いつ物を持っていく必要があるかをシステムに教える手間」である．自分では理解していることであっても，それを体系的にシステムに支持するのは案外大きな手間だ．例えば，ノート PC 本体は毎日会社に持っていかなければならないが，充電器は出張の時だけ持っていく必要があり，ディスプレイ出力アダプタはこの会議のあるときに……などという登録を，誰がやりたいと言うだろうか．さらに，これも環境の変化や季節の移り変わりなどによって変化しうるのである．やはり，これを自力で登録するのは無理がある．先に紹介した Hui-Huang Hsua らの研究や照屋のぞみらの研究など，この点に着目した論文も存在したが，曜日で固定されていれば可能という程度であり，様々な行動パターンを示す人々の多くをカバーできるとは言い難い．

これらの問題を解決できれば，忘れ物に対する有効な解決策になりうるのではないだろうか．本研究は，この 2 点に対する解決策を提案するものである．

第3章 システム設計

この章では，提案するシステムのおおまかな設計を行う．

3.1. RFID システムにおけるセンシングデータ

RFID を用いたシステムにおいては，データの信頼性の無さを考慮することが欠かせない．RFID は無線通信に基づくものであり，電波干渉などの不確定要素が多いためである[45],[46],[47]．現実世界では，RFID のリードレートは 60〜70%だとも言われている[48]．この問題への標準的な対応策は，平滑化フィルタをかけることである[49]．これは RFID リーダから得られたストリーミングデータに対してウィンドウを設定し，このウィンドウを時間的にスライドさせることで，補完を行うものである（図 3.1）．

この手法においては，設定するウィンドウのサイズが非常に重要である．大きなウィンドウを利用すれば偽陽性（RF タグが検出範囲外に出たにも関わらず，継続してエリア内に存在すると判断してしまう）が，小さなウィンドウを利用すれば偽陰性（RF タグが検出範囲内にあるにも関わらず，エリア外だと判断してしまう）を引き起こしやすいためである．そこで，Shawn R. Jeffery ら[50]は，このウィンドウのサイズを可変にし，受信状況に応じて拡大・縮小する SMURF 演算という手法を提案している．この手法では，ごく短時間（約 0.2〜0.25 秒）の受信データを「エポック」として纏め，このエポック中においてリーダから何回問いかけたうち何回応答があったかという情報を使ってウィンドウサイズを変更する．例えば，10 回問いかけたうち 9 回に応答があればリーダの近くにいると考えてウィンドウサイズを縮小し，2 回しか応答がなければウィンドウサイズを拡大するといった形である．

先にも紹介した Tiancheng Zhang ら[44]の物の包含関係を捉えようとする研究ほか，非常に多くの研究でこの SMURTF 演算は使用されている．本研究においてもこの SMURF 演算の適応を検討したが，3 つの理由から今回の採用を見送った．

まず 1 つめの理由が，今回研究に利用した RFID リーダの仕様である．SDK（SoftwareDevelopmentKit）を用いたリーダの使用方法については，詳しく 4.2.1 節で調べるが，このリーダは何回 RF タグに呼びかけたのかという情報を提供しない．これは，棚卸などといった用途では呼びかけの回数といった情報は不要であるため，SDK から機能が省かれているものと想定される．他社製品を用いることについても検討を行ったが，今後普及する RFID リーダでも同様にこの機能が省かれている可能性が否定しきれないため，より基本的な機能のみでシステムを利用できたほうが良いと考えた．

もう 1 つの理由が，本システムがリーダに対して求める要件とのバランスである．本研究で重要なことは，リーダの検出領域を物が出入りするタイミングを正確に把握することではない．重要なことは，家を出るその時に，物を持っているか正確に把握することにある．また，リーダと RF タグの距離はあまり変化しない．また，本研究において偽陰性は「忘れ物アラートが過剰に出てしまう」ことだと言って差し支えなく，偽陽性は「忘れ物アラートが出るタイミングが遅れる」ことだと言える．アラートが遅れることも問題ではあるが，過剰なアラートはユーザのアラート軽視へと繋がるものであり，こちらをより重視して削減すべきだと考えた．

最後の理由は，システムにおける計算量である．本研究の想定では，リーダは常に動作し，スマートフォンも常に通信を続ける．サイズが固定されたウィンドウを適応する程度であれば比較的処理量は抑えられるが，動的に変更するとなると計算量もかなり増える．エポックと呼ばれる通信の最小単位ごとの計算量を比較すると，ウィンドウサイズが固定されていれば 1 つの if 文で済むところを，SMURF では図 3.2 のような計算を行わなければならない．RF タグ 1 つだけであれば大した問題ではないが，システムが管理する RF タグの個数が増えれば一定量の計算量になることが想定される．もちろん近年の高性能なスマートフォンにおいて処理が追いつかないほどではないが，処理の増加はバッテリ消費の増加とスマートフォン全体のユーザエクスペリエンスの低下に繋がる．削減できる部分については，削減できた方が良いのは言うまでもない．

以上により，本研究ではサイズを固定したウィンドウを使用して，RFID ストリーミングデータを平滑化することとした．なお，ウィンドウサイズの検討については，5.1.2 節にて行う．

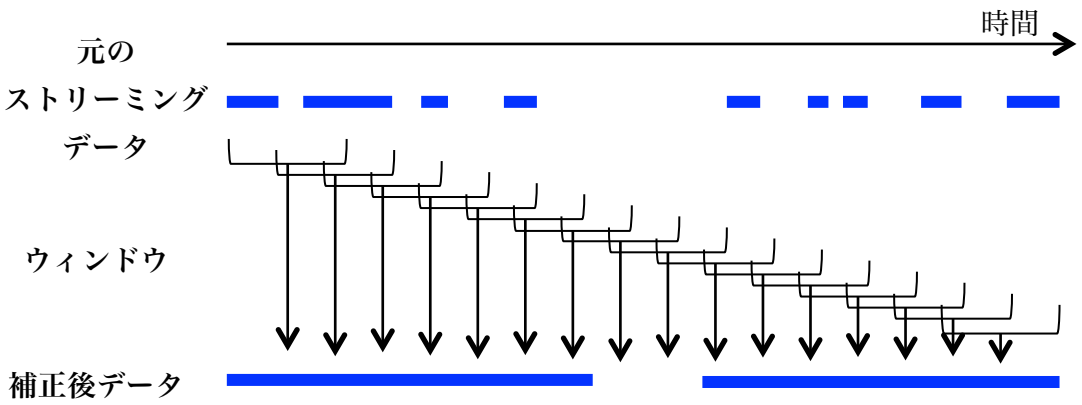


図 3.1　ウィンドウによるデータの平滑化．

```
Require: T = set of all observed tag IDs
         δ = required completeness confidence
∀i ∈ T, w_i ← 1
while (getNextEpoch()) do
  for (i in T) do
    processWindow(W_i)
    w_i^* ← completeSize(p_i^{avg}, δ)              // Lemma 4.1
    if (w_i^* > w_i) then
      w_i ← max{min{w_i + 2, w_i^*}, 1}
    else if (detectTransition(|S_i|, w_i, p_i^{avg})) then
      w_i ← max{min{w_i/2, w_i^*}, 1}
    end if
  end for
end while
```

図 3.2　SMURF 演算におけるタグ毎クリーニングアルゴリズムを擬似言語で示したもの．（出典：Adaptive Cleaning for RFID Data Streams[50]）

3.2. ユーザが必要とする持ち物の推定

これまでも議論してきたように，本研究では RFID とユーザが物名称の対応表を作成したり，持ち物と事象の関係性を登録したりといったことをしない．これを実現するためには，過去の持ち出しログからユーザが必要とする持ち物（の RFID）を推定することが必要である．しかし，2.1 節で議論したように，事象と持ち物は一方向的対応であるため，単純に推定することはできない．

本節では，この問題に対するベイズ統計学的アプローチを提案する．

3.2.1. 物・事象関係性推定アルゴリズムにおける入出力データ

物–事象関係性推測アルゴリズムについて説明する前に，そもそもどのようなデータが入力され，どのようなデータの出力を期待するのかについて議論する．第 2 章でも議論したように，システムが物と事象の関係性を推測するには，どのような事象があった日にどのような物が持ちだされたのかというログを収集する必要がある．

まず，各日に持ち出した物を記録することについて検討する．これまでも議論してきたように，本システムは RFID リーダを持ち歩いていることを前提にしているため，

携帯するリーダがリアルタイムに持ち物のログを収集可能である．一般的に，RFID リーダは受信したタグ ID と同時に受信電波強度を出力することもできる．しかし，電波強度は環境に影響を受けやすいことに加え，UHF 帯の電波は直進性が強いためにリーダの指向性が強い傾向があり，明確にリーダを物に向けて使わない本研究のような使い方では，電波強度を測距に使ったりすることは現実的ではない．そのため，本研究では RF タグの ID のみを取得する．また，リーダから得られた RFID ストリーミングデータは，◯時◯分◯秒に ID：◯◯のタグと通信した，という点のデータの連続であるため，これを 3.1 節で説明した平滑化処理を利用して，◯時◯分◯秒〜×時×分×秒まで ID:◯◯のタグと通信した，という時間的継続性のあるデータに整理する．

続いて，その日の事象の収集についてであるが，これはユーザの手間を極力省くために，外部から取得するものとした．事象として考えられるものとしては，予定のほかに晴天・雨天や気温，花粉飛散状況，季節，気温といったものがあるが，これらはいずれもインターネットから取得が可能なものである．例えば Google 社のオンラインカレンダーサービス Google Calendar では，公式に提供されている API[51]を用いることで指定日の予定を取得したりすることが可能である．気象関係についても，LINE 社の提供する Livedoor Weather Hacks[52]をはじめとして，各社が有料・無料の API を提供している．これら API を使用することで，ユーザに特別な操作を求めることなく，必要な事象を取得することができる．また，更に事象の種類を追加したい場合には，それを取得できる API を使ったアドオンを作成することで，対応が可能であると考えられる．

このようにして収集したデータは表 3.1 のようなものである．これが物–事象関係性推測アルゴリズムの入力データとなる．当然のことだが，この段階ではどの事象でどの RF タグ（が貼付された持ち物）が必要であるかはわからない．これを用い，表 3.2 のようなデータを出力するのが物–事象関係性推測アルゴリズムに期待されることであるが，2.1 節で議論したように，事象と物には一方向的対応性があるため，解析的に解いて表 3.2 の内容を確定させることはできない．

表 3.1　システムに入力されるデータの例．

日付	その日にあった事象	その日持ちだされた RF タグ
2014/1/26	雨，論文提出，ゼミ，MTG	001，005，006，007，008，011
2014/1/27	晴，健康診断	009
2014/1/28	晴，ゼミ，発表，MTG	004，005，006，007，008，
2014/1/29	雨，輪講，MTG	001，005，007，008
⋮	⋮	⋮

表 3.2　システムに期待する出力データの例.

事象	晴	雨	論文提出	ゼミ	MTG	健康診断	発表	輪講
必要な RF タグ	(なし)	001	011	005 006 008	005 007	009	004	005 008

3.2.2.　予定名におけるタギング

前節でも論じたように，事象は数多くの種類が考えられる．しかし，その中でも特に注目が必要なものは予定である．日々存在するという意味では，気象に関係する事象も日々現れるものであり，重要度が高いと言える．しかし，予定はユーザに自由記述を許す点で，気象関連事象とは全く異種のものである．

ユーザが予定名を自由に記述できるということは，物–事象関係性推測アルゴリズムにおける推測をかなり困難にする．例えば,「西 10 号館 6 階 T 研究室ゼミ室で○○さんたちと A プロジェクトについて話し合う」という予定と「西 3 号館 2 階 I 研究室ゼミ室で××さんと A プロジェクトについて調整する」という予定があったとする．人間が見れば，これは同種の予定であるとわかる．持ち物もおそらく似通ったものになると推測できる．しかし，システムから見れば全く異なるものである．このような名前をつけた予定ばかりでは，当日の持ち物を推測するのに使えるログデータを充分に収集するのはかなり困難だと言わざるを得ない．

本来，これに対する解決としては，システムが予定名称の意味を理解し，同種の予定であると判断できることが望ましい．本研究はユーザに特別な準備や対応を求めないことが原点にあるためである．しかし，現実問題としてこれはかなり難しいと言わざるを得ない．言語を理解した上で，そのユーザの癖なども知る必要があると考えられるためである．

また，似ていつつも，別の予定であることにも注目しなければならない．先の例で言えば，西 10 号館で話し合う時とは異なり，西 3 号館の部屋に入るためにはカードキーを用意する必要があるかもしれない．これまでユーザが一度も西 3 号館入ったことがなければこれを推測するのは理論上不可能である．しかし，過去に別の予定であっても西 3 号館に入ったことがあれば，このカードキーについてもシステムが把握していることが期待される．

これら両方を満たすためには，形態素解析を行って名詞を選り出すといった方法が考えられる．言わば，選り出した名詞 1 つ 1 つを事象として扱うのである．しかし，これでもまだノイズが多すぎるだろう．ノイズの多さは，正確な推測の妨げになる．

そこで，今回，ユーザの予定名入力に際し，ある程度のルールを設けることで持ち物と事象の関係性推測を手伝ってもらうこととした．そのルールが,「予定タグ」である．これは，予めユーザに予定のジャンル分けを行ってもらうことに等しい．先の例で言えば,「【A プロジェクト】【T 研ゼミ室】【MTG】○○さんと話し合い」「【A プロジェクト】【I 研ゼミ室】××さんと調整」といった形である．これであれば，システムも容易に類似の予定であることを理解できる．この予定タグの付け方は自由であり，どのような付け方をするかはユーザに一任する．しかし，予定タグというルールを作ることで，予定名の付け方におけるぶれを軽減できることが期待できる．

3.2.3.　物−事象関係性推測アルゴリズム

本節では，本研究の要の 1 つである，物と事象の関係性を推測する手法について議論する．2.6.3 節で論じたように，既存研究で同様な推定に取り組んでいるものはなかった（最も近い研究としては Tiancheng Zhang ら[44]のものがあるが，これは物と物の関係性を捉えようとするものである）．

・事象と物の共起確率を求める手法

ここでは，それぞれの事象と物の組み合わせについて，共起確率（条件付き確率）を求めるアルゴリズムを提案する．例として，物が 3 つ（a〜c），事象も同じく 3 つ（A〜C）の場合について考える．それぞれの物がそれぞれの事象と関係している可能性があるため，これを表に纏めると，表 3.3 のようになる．なお，$P(Y|X)$は事象X（原因）があったときに物Y（結果）が存在する確率を示し，以下のように求めることができる．

$$P(Y|X)=\frac{\textit{事象X があった日のうち，物Y が持ちだされた日}}{\textit{事象X があった日の合計回数}} \tag{3.1}$$

ここで，システムの利用開始からt日目において，

- x_t：t日目において事象Xがあった場合には 1，なかった場合には 0
- $P_t(Y|X)$： t日目における$P(Y|X)$
- y_t：t日目において物Yを持ちだされた場合には 1，持ちだされなかった場合には 0

とすると，

$$P_t(Y|X)=\frac{\sum_{n=1}^{t} x_n y_n}{\sum_{n=1}^{t} x_n} \tag{3.2}$$

以上のよう書くことができる．

また，$P_{t+1}(Y|X)$は次式のように書くことができる．

$$P_{t+1}(Y|X) = \frac{\sum_{n=1}^{t+1} x_n y_n}{\sum_{n=1}^{t+1} x_n} = \frac{\sum_{n=1}^{t} x_n y_n + x_{t+1} y_{t+1}}{\sum_{n=1}^{t} x_n + x_{t+1}} \tag{3.3}$$

表 3.3　事象・物ともに 3 つである場合におけるそれぞれの共起確率.

		物		
		a	b	c
事象	A	$P(a\|A)$	$P(b\|A)$	$P(c\|A)$
	B	$P(a\|B)$	$P(b\|B)$	$P(c\|B)$
	C	$P(a\|C)$	$P(b\|C)$	$P(c\|C)$

・事象と物の共起確率を求める手法にタグの考え方を追加する

(3.1)式のように計算することで，事象と RF タグの関係性を共起確率で求めることができるが, このままでは 3.2.2 節で議論したタギングが考慮されていない. しかし，持ち物へ影響を与えるという意味では，事象も予定タグも同列に扱うことができる．

そこで，先に定義した$X, P(Y|X), x_t$について，それぞれ以下のように拡張する．

- X：事象もしくは予定タグ
- $P(Y|X)$は事象もしくはタグXがあったときに物Yがおきる確率
- x_t：t日目において事象もしくはタグXがあった場合には 1，なかった場合には 0

・事象と物の共起確率を求める手法において時間経過を考慮する

式(3.2)・式(3.3)で示したような手法では，直近のデータでも 1 年前のデータであっても同列に扱われる．しかし，持ち物は時間経過によって変化しうる．例えば，3 月までは会社で ProjectX の仕事をしていたためその資料を会社に持って行っていたが, プロジェクト終了に伴い 4 月からは持っていく必要がなくなった，という場合に，4 月以降もアラートが出続けていては困るだろう．やはり，古いデータに比べて新しいデータはより信頼のおける，重視すべきデータだと言える．

そこで，新規に重要度減衰パラメータ$0 < k \leq 1$を定義する．このパラメータkはt日目データに対する$t-1$日目データの重要度を示す．すなわち，$t-1$日目データはt日目データのk倍の価値しかない，ということを示す．

これを式(3.2)と式(3.3)に適用すると，以下のように書くことができる．

$$P_t(Y|X) = \frac{\sum_{n=1}^{t} x_n y_n k^{t-n}}{\sum_{n=1}^{t} x_n \, k^{t-n}} \tag{3.4}$$

$$P_{t+1}(Y|X) = \frac{(\sum_{n=1}^{t} x_n y_n k^{t-n})k + x_{t+1}y_{t+1}}{(\sum_{n=1}^{t} x_n k^{t-n})k + x_{t+1}} \tag{3.5}$$

例えば，$k = 0.99$とすると，30 日後におけるデータの価値は 0.74 となる．kが小さすぎると新しいデータに引っ張られてしまい，1 日例外的な日があっただけで状況が全く変わってしまうが，kが大きすぎると持ち物の変化に追従することができない．この値については，5.2.2 節で示す実験にて検討する．

3.2.4.　必要な持ち物の推定

3.2.3 節で提案した式により，事象と物の関係性を確率という形で求めることができるようになった．生成された関係性の表と当日に存在する事象のリストがあれば，見比べることで必要な持ち物が推定できる．しかし，そのためには閾値をどのように設定するかを考える必要がある．

提案した物–事象関係性推測演算は，条件付き確率による相関分析の派生ということができる．そこで，相関分析における慣用表現である「強い相関」「中程度の相関」「弱い相関」を推測演算の信頼度に当てはめ，なにが必要な持ち物であるかの判定に用いることとした．具体的には，

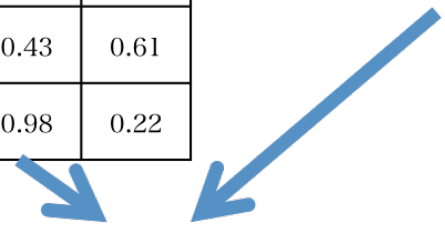
・ 1.0～0.7：強い相関→High
・ 0.7～0.4：中程度の相関→Middle
・ 0.4～0.2：弱い相関→Low
・ 0.2～0.0：ほとんど相関がない→Zero（通知なし）

という閾値を判定基準に用いている．

この High/Middle/Low はそのままユーザインタフェイスに反映するものとし，今回ユーザへの通知に利用するスマートウォッチにおいて，通知を出すときの背景色に反映するものとした．このユーザインタフェイスについて，詳しくは 4.2.7 節で議論する．また，同じ物が同日の複数の事象によって必要であると判定された場合，最も高い通知レベルを使用して通知される（例えば，図 3.3 の物 e を忘れた場合，今日の持ち物リストの上から 4～6 番目が該当するが，通知では Middle が使用される）．

		モノ				
		a	b	c	d	e
事象・予定タグ	A	0.55	0.21	0.15	0.22	0.37
	B	0.06	0.04	0.16	0.66	0.54
	C	0.04	0.93	0.24	0.77	0.53
	D	0.64	0.19	0.12	0.43	0.61
	E	0.42	0.14	0.11	0.98	0.22

今日の事象	A, D, E



今日の持ちモノ		
モノ	必要な事象・予定タグ	信頼度 (通知レベル)
d	E	High
a	A	Middle
a	D	Middle
e	D	Middle
e	A	Low
e	E	Low
b	A	Low

図 3.3　必要な持ち物の推定イメージ.

3.2.5.　ユーザからのフィードバック

今回提案したアルゴリズムでは, 必ずしも正確に持ち物を推定できるとは限らない. 例えば, いつも「ゼミ」と「輪講」が同日中にあり, その日はいつも「タブレット PC」を持って行っていたとする. ある日,「ゼミ」がなくて「輪講」だけがあったとすると,「タブレット PC」が必要であるかどうか, アルゴリズムで確定させることは可能だろうか. 不可能である. 式(3.4)のように計算を重ねても, $P(タブレットPC|ゼミ)$と$P(タブレットPC|輪講)$は共に高い値を示すことだろう.

この問題を解決するには, ユーザからフィードバックする仕組みがかかせない. 最も単純なのは, 表 3.3 における該当セルに対して×をつけてしまうという方法である.

すなわち，$P\left(タブレットPC \middle| 輪講\right) = 0$とし，今後「輪講」があった日に「タブレットPC」を持ちだされても値を更新しないという方法である．

しかし，この手法では，元に戻すことはできなくなってしまう．しばらく必要なかったがまた必要になった，というような状況を学習することもユーザから教えることもできない．そこで，tを以下のように拡張する．

・$t_{Y,X}$：事象もしくは予定タグXと物Yについて，ユーザからフィードバックが与えられた日，もしくはシステムの利用開始日からの日数

すなわち，ユーザからフィードバックを与えられた時点で，$t = 0$に巻き戻すと同時に$P\left(タブレットPC \middle| 輪講\right) = 0$とする．これまでに蓄積されたデータを破棄するという仕組みである．

3.3. 外出の検出

3.2 節で提案したアルゴリズムによって，ユーザが必要とする持ち物を推測することが可能になった．しかし，大事なことは家の中で収集したデータを演算に含めてはならない，ということである．家の中で収集されたデータは，その日に持ち出さない物の RFID が含まれている可能性がある．

また，家の中ではいくら忘れ物という判定がなされても，この通知が出てはならない．常に出続けている通知は，ユーザに軽視されることに繋がる．必要な時に，必要な通知が出ていることが大事なのである．

以上により，在宅中か外出中かを検出することが欠かせないと言える．この検出をするには様々な方法が考えられるが，今回は玄関に専用 RF タグを貼付するという方法を選択することにした．本論文では，この専用 RF タグを玄関タグと呼ぶこととする．この方法の場合，自宅にいる間も常に RFID リーダと通信を続け，RFID データを取得し続ける．そして，玄関タグの RFID を受信した場合，在宅モードと外出モードを切り替える．

この手法を使う場合に考えるべきことは，今が在宅なのか外出中なのか，その判断をする方法と，在宅中に RF タグを読み込んでしまったら誤った認識をされるようになってしまうのではないか，ということである．

前者については，単純に，管理対象の RF タグを登録した場所が自宅，と判断することにした．詳しくは 4.2.7 節で議論するが，システムが 1 度でも通信したことのある RF タグ全てについて物−事象関係性推測演算を行っていては，不必要な演算が膨大に発生する可能性がある．このため，管理対象の RF タグを登録するモードを設ける．

このモードの時，自宅にいるとして記録する．以降は玄関タグを読み込む度にモードを切り替えれば良い．

後者については，運用の方法で充分に対応可能である．例えば，玄関の外側に貼付すれば，在宅中に読み込むことはまず考えられない．一軒家であれば，ポストの下に貼付しておくなどといった方法も考えられる．また，万が一狂ってしまった場合には，在宅中に Management モードを一旦 ON にするだけで修復されるので，解決は容易である．

第4章 システムの実装

本章では，ここまで論じてきたシステムを実装し，提案システムの実現性を確認する．まず今回使用する機材類について説明した後，どのような構造になっているかを説明する．

4.1. 使用する機材

4.1.1. RFID システム

本節では，評価用のシステムで用いた RFID システムについて説明する．

今回，ユーザがセンシングのため持ち歩く RFID リーダとして，東北システムズサポート社の DOTR-910J[53]を使用した．図 4.1 に DOTR-910J の写真，表 4.1 に DOTR-910J のスペックを示す．このスペックシートにある通り，DOTR-910J はバッテリを内蔵しており，人が容易に持ち歩くことが可能である．

また, DOTR-910J には, 同機をコントロールするための Android 用及び Windows 用の SDK が付属している．この SDK については，4.2.1 節で説明する．

図 4.1　東北システムズサポート社のモバイル RFID リーダ，DOTR-910J.

表 4.1　DOTR-910J のスペック.

項目	仕様
周波数	916.8MHz〜923.4MHz
送信出力	250mW，24dBm
対応プロトコル	ISO 18000-6C, EPCglobal Class1 Gen2
アンテナ	内蔵
偏波特性	円偏波
インタフェイス	Bluetooth Class2
バッテリ	2350mAh 充電式リチウムポリマーバッテリー
電源方式	USB

4.1.2.　スマートフォン・スマートウォッチ

4.1.1 節で論じたように，今回使用する RFID リーダには Android 用と Windows 用が付属している．そこで，今回は Android 端末を用意することとした．また，それに合わせて，スマートウォッチとして Android Wear 端末を使用する．

用意した端末は，どちらも LG 電子社の製品で，nexus 5 と LG G Watch R である．製品の写真は図 4.2 に，スペックはそれぞれ表 4.2 と表 4.3 に示した．

図 4.2　nexus 5 と LG G Watch R.

表 4.2　nexus 5 のスペックシート.

項目	仕様
メーカ	LG 電子
機種名	nexus 5
型番	LG-D821
Bluetooth	4.0
連続待受時間	300 時間
OS	Android 4.4.4

表 4.3　LG G Watch R のスペックシート.

項目	仕様
メーカ	LG 電子
機種名	LG G Watch R
型番	LG-W110
Bluetooth	4.0
連続稼働時間	約 1 日
OS	Android Wear 5.0.1
画面	円形 P-OLED 液晶　1.3 インチ

4.2. 実装

2.5 節で議論したシステム構成について，さらに細かく見ていくと図 4.3 のようになる．背景に色がついている部分がスマートフォン内部で処理される部分である．

予定表もスマートフォン内部として描かれているのは，Android の機能として Google カレンダーと端末内部のカレンダー（予定表）を同期する機能があり，Google カレンダー→端末内部のカレンダー→提案システムという流れで予定を取得することが可能であるためである．一方，天気予報は API を用いてインターネットから取得するものであるため，背景色がついている部分から外れている．

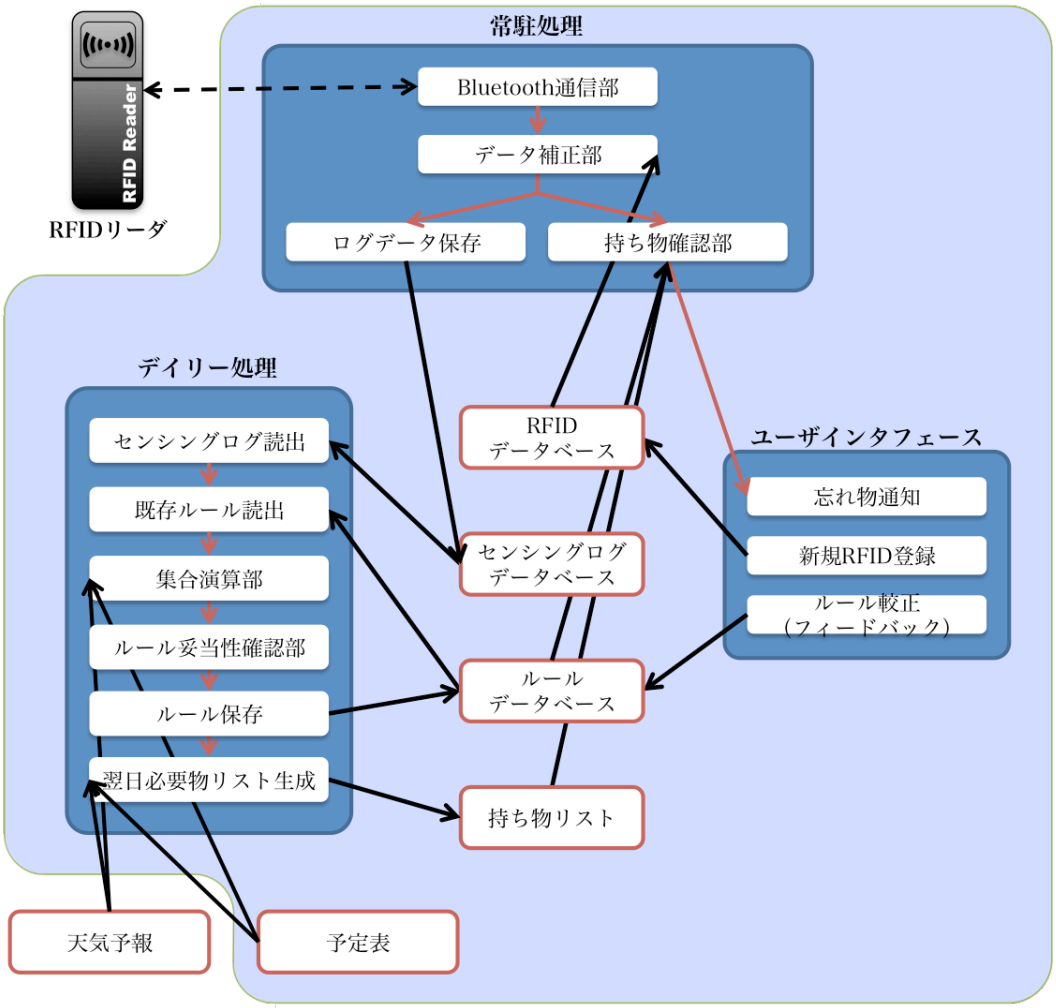


図 4.3　システムの詳細な構成図.

4.2.1. RFID リーダの Android 用 SDK

先に論じたように, 今回使用する RFID には開発用 SDK が付属している. この SDK について，本研究に関係のある部分についてのみ，簡単に説明する．なお，本 SDK はリーダのメーカである東北システムズサポート社より提供を受けたものである.

利用のおおまかな流れは図 4.4 のとおりである．このリーダはスマートフォンと SPP（Serial Port Profile）を用いて通信を行っているが，シリアル通信に関わる部分は全て SDK 側でラッピングされている．そのため，開発者側でシリアル通信に関することを意識する必要はなく，connect メソッドを利用する際にリーダの MAC アドレスを指定することでリーダとコネクションが張られる．しかし，Bluetooth を利用するにあたって必須となるペアリングに関してはその機能が提供されていないため，開発者側で実装するなりユーザにペアリングを求める仕組みなりが必要となる.

設定に関しては，同一 RF タグが複数回読み込まれた場合の扱いであるとか，電波の照射，アンチコリジョン機能の調整などといったことが必要となる．設定項目により，電源切断後もその設定内容が保存されるものとリセットされるものがあるが，別アプリケーションで設定が変更されている可能性を考慮し，今回は動作に必要な設定の全てを毎回行うこととした.

また，このリーダにおいて，読み込みは 2 種類のモードがあるが，今回は inventoryTag メソッドを使うものとした. このモードでは, リーダは RF タグの EPC 領域（タグの ID のこと）及び設定によって受信時刻・受信電波強度を得ることができる．今回の場合，3.2.1 節で論じたように受信電波強度は利用せず，また受信時刻についてもスマートフォンの時計を利用すれば良いことから, ID のみを受信している.

この inventoryTag メソッドを使用してから stop メソッドを呼びだす間に，リーダが RF タグを発見すると，onInventoryEPC イベントが発生する．このイベントを setOnDotrEventListener でセットしたリスナーで受け取ることにより，システムが RFID を利用することができる.

この他にも様々なメソッドやイベントが存在し，適宜それらメソッドを呼び出したりイベントを受け取ったりすることで操作を行う.

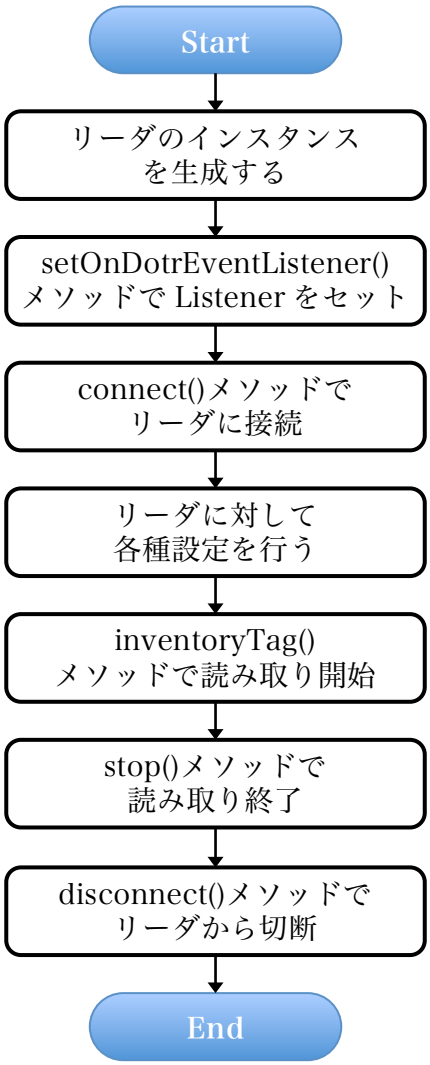


図 4.4　SDK によるリーダ利用の流れ.

4.2.2. MainActivity

本アプリケーションにおいて，最初に実行されるのがこの部分である．ソースコードの一部をソースコード 4.1 に示す．Android において Activity は 1 つの画面に相当する最も基本的なコンポーネントであり，アプリ起動時の動作を規定したり，トップ画面を生成したりするのがその役割である．

開発したアプリにおいて，MainActivity は枠のみを提供し，枠の内部は Fragment というコンポーネントを読み出すようになっている（図 4.5）．

Android アプリにおける Activity には，定められたライフサイクルがある（図 4.6）．

このうち，MainActivity では onCreate()と onStart()を利用している．onCreate()はアプリの起動時に呼ばれるものであるので，まず MainActivity の枠の中に表示する MainFragment の呼び出しを行う．

その後，Broadcast Receiver というものの準備を行う（13 行目〜）．これはアプリ内もしくはアプリ外との連携動作をとるための仕組みであり，ここで指定した Broadcast intent を他のアプリ等が発行すると，この MainActivity に通知がくるというものである．今回は，後述する ReaderService からリーダとの接続状況受け取りと Android の AlertManager からの通知受け取りに使用している（19 行目〜）．前者については，Intent を受け取ったら MainFragment に Listener という仕組みを使ってそれを通知し，画面に反映するようになっている．これは Fragment では Broadcast を直接受け取ることができない（Fragment は Activity の一部品という考え方であるため）ので，このように一旦 Activity を経由するようになっている．後者の AlertManager は 1 日 1 回物–事象関係性推測演算を行うために必要なもので，後述の設定画面で指定された時刻になったら Android から通知を受け取るためのものである．なお，ここで準備した Broadcast を受け取った場合の処理を規定しているのは，39 行目〜の部分である．

続いて onResume()では，Bluetooth アダプタの On/Off を確認する．Off だった場合には，ユーザに確認をとって On にする．これを onCreate でやらないのは，このアプリがバックグラウンドにある間にユーザが Bluetooth を Off にしてしまった場合に，それを検出するためである．

メニューの表示も MainActivity の役割である．図 4.5 の右上に出ているようなメニューを表示し，どれかのメニューが選択された場合に以下の処理を実行する：

- メイン画面：MainFragment に画面遷移する
- 環境設定：PrefsFragment に画面遷移する
- このアプリについて：アプリ情報と Copyright を表示するダイアログを出す
- DB 書き出し：主にデバッグに使うためのものであり，SQLite ファイルを外部記憶領域に保存する

これらへの画面遷移については，changeFragment()というメソッドを作成し，より容易に遷移ができるようにした．また，ダイアログについては CommonDialogFragment というアプリ内共通で使うことのできるダイアログを作成した．このダイアログについては，4.2.6 節にて説明する．

MainActivity 内の処理のうち，一部をソースコード 4.1 に示す．39 行目以降が，先に述べた Broadcast intent を受け取った場合の処理に相当する．

図 4.5　Activity と Fragment の関係.

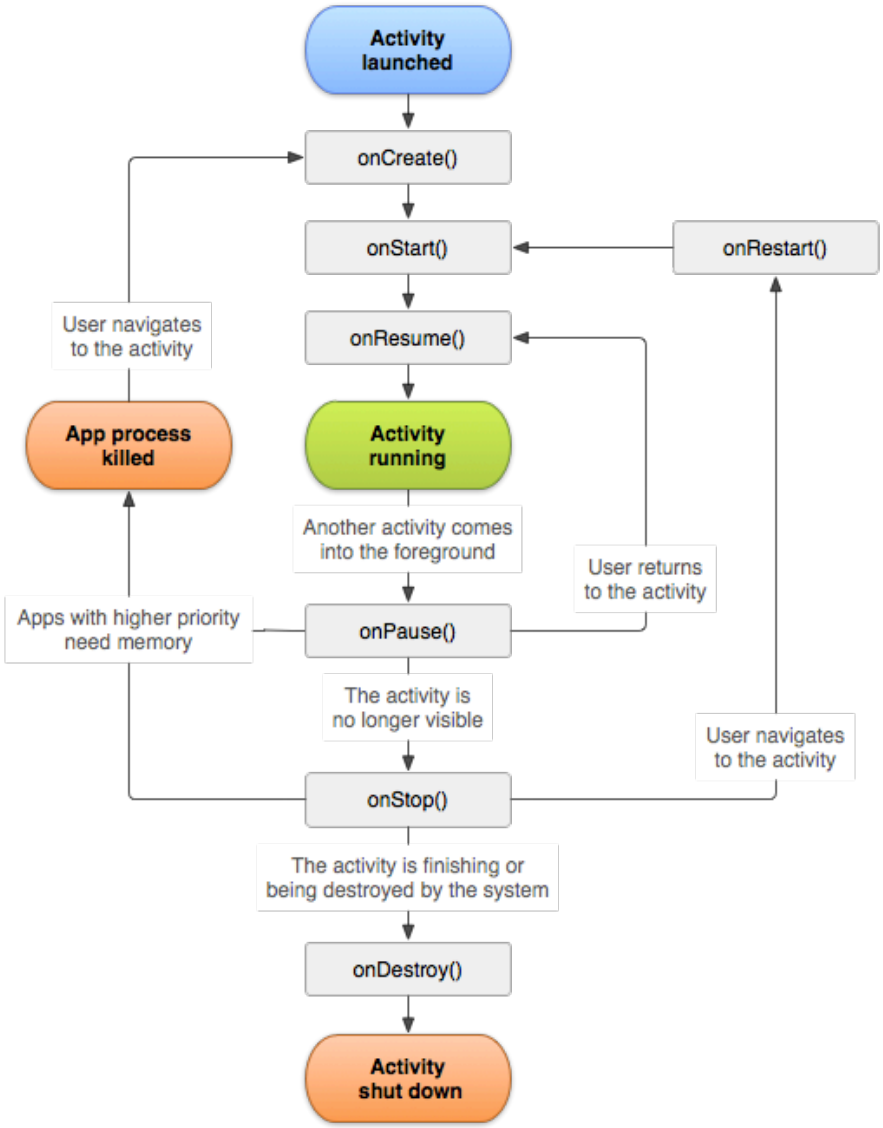


図 4.6　Android における Activity のライフサイクル.（出典：Android Developers[54]）

```
1     @Override
2     protected void onCreate(Bundle savedInstanceState) {
3          super.onCreate(savedInstanceState);
4          pref = PreferenceManager.getDefaultSharedPreferences(this);
5
6          setContentView(R.layout.activity_main);
7          if (savedInstanceState == null) {
8               Fragment mainFragment = new MainFragment();
9               getFragmentManager().beginTransaction()
10                        .replace(R.id.container, mainFragment).commit();
11         }
12
13 //        BroadcastReceiver の準備
14         intentFilter = new IntentFilter();
15         intentFilter.addAction(BROADCAST_READERSERVICE_STATUS);
16         intentFilter.addAction(BROADCAST_PURESUME_ALARM);
17         registerReceiver(broadcastReceiver, intentFilter);
18
19 //      AlarmManager で指定時刻に立ち上げてもらう準備
20         Intent intent = new Intent(getApplicationContext(), MainActivity.class);
21         intent.setAction(BROADCAST_PURESUME_ALARM);
22         PendingIntent sender = PendingIntent.getBroadcast(this, 0, intent, 0);
23         AlarmManager am = (AlarmManager)(this.getSystemService(ALARM_SERVICE));
24         Calendar calendar = Calendar.getInstance();
25         Date today = calendar.getTime();
26         calendar.set(Calendar.HOUR,pref.getInt("puresume_time_Hour",
                    MainActivity.PURESUME_DEFAULT_HOUR));
27         calendar.set(Calendar.MINUTE, pref.getInt("puresume_time_Minute",
                    MainActivity.PURESUME_DEFAULT_MINUTE));
28         Date setDay = calendar.getTime();
29         if(today.after(setDay)){
30              calendar.add(Calendar.DATE, 1);
31         }
32         Log.d(TAG_FOR_LOG, "AlarmFirstSetTime:" + calendar.toString());
33         am.setInexactRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(),
                    AlarmManager.INTERVAL_DAY, sender);
34
35 //        Bluetooth アダプタのインスタンスを取得しておく
36         readerManager = ReaderManager.getInstance(this);
37    }
38
39    /**
40     * ReaderService などからの broadcast を受け取るためのクラス
41     */
42    private final BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
43         @Override
44         public void onReceive(Context context, Intent intent) {
45              if(intent.getAction().equals(BROADCAST_READERSERVICE_STATUS)) {
46                   Bundle bundle = intent.getExtras();
47                   int readerServiceStatus = bundle.getInt("status");
48
49                   Log.d(TAG_FOR_LOG,
                              "Receive Broadcast:readerServiceStatus("+ readerServiceStatus +")");
50                   listener.readerServiceStatusChange(readerServiceStatus);
51              }else if(intent.getAction().equals(BROADCAST_PURESUME_ALARM)
                         && pref.getBoolean("presume_alert",false)) {
52
53 //                  推測演算をおこなう service をたちあげる
54                   Intent serviceIntent = new Intent(MainActivity.this, PresumeService.class);
55                   startService(serviceIntent);
56              }
57         }
58    };
```

ソースコード 4.1　MainActivity における処理の一部.

4.2.3. MainFragment

前節で論じたように, MainActivity が立ち上げられた際に最初に表示されるのがこの MainFragment である（ソースコード 4.2）. Fragment は画面上の 1 つのコンポーネントであり, 実装の仕方次第では 1 つの Activity 上で 2 つ以上の Fragment が同時に表示されることも考えられるが, 今回のアプリでは常に単独で表示される.

MainFragment の画面は非常に単純で, トグルスイッチが 2 つ表示されているだけである（図 4.7）. 上のスイッチを On/Off と切り替えると, RFID リーダとの接続・センシングをはじめとする常駐処理を開始したり停止したりすることができる. これらの処理は, 後述する ReaderService によって行われる. ソースコードの 35 行目〜の処理は, MainFragment が描画される際, 既に RFID リーダと接続されていた場合に, スイッチの状態を予め On にしておくためのものである. この処理をしなければ, 現在リーダと接続しているのかユーザ側から把握することが出来ない.

下のスイッチを On/Off した場合には, 物–事象関係性推測演算を行うかどうかを切り替えることができる. この処理は PresumeService によって行われるが, ReaderService とは異なり, スイッチを On にした瞬間に Service が立ち上がるわけではない. AlertManager から Broadcast が飛んできた場合（4.2.2 節参照）に, PresumeService が立ち上がる. このスイッチが On の場合のみ, PresumeService にて持ち物リストが生成されるのである.

ソースコード 4.2 に示したソースコードは, MainFragment クラスの全てであるが, MainFragment は MainActivity クラスの中に存在するため, MainActivity 内の様々なメソッドと連携しているため, このコードのみでできているわけではない.

図 4.7　MainFragment における画面の変化．

```
1     public static class MainFragment extends Fragment {
2          public final String TAG_FOR_LOG = "MainFragment";
3
4          View rootView;
5          SharedPreferences pref;
6
7          @Override
8          public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                    Bundle savedInstanceState) {
9               Log.d(TAG_FOR_LOG,"onCreateView");
10              rootView = inflater.inflate(R.layout.fragment_main, container, false);
11              pref = PreferenceManager.getDefaultSharedPreferences(getActivity());
12
13 //             service の状態によってリーダ接続スイッチの ON/OFF を切り替える
14              setReaderSwitch(pref.getInt("ReaderService_status",
                          MainActivity.SERVICE_STATE_STOP));
15              Log.d("debug","pref:" + pref.getInt("ReaderService_status",
                          MainActivity.SERVICE_STATE_STOP));
16
17              TextView modeText = (TextView)rootView.findViewById(R.id.now_mode);
18              if(pref.getBoolean("management_tag_collect", false)){
19                   modeText.setVisibility(View.VISIBLE);
20              }else{
21                   modeText.setVisibility(View.INVISIBLE);
22              }
23
24 //           リスナーのセット
25              ((MainActivity) getActivity()).setReceiverListener(new receiverListener() {
26                   @Override
27                   public void readerServiceStatusChange(int readerServiceStatus) {
28                        setReaderSwitch(readerServiceStatus);
29                   }
30              });
31
32              return rootView;
33         }
34
35         public void setReaderSwitch(int state){
36              final Switch readerSwitch = (Switch)rootView.findViewById(R.id.reader_switch);
37              switch (state){
38                   case SERVICE_STATE_STOP:
39                        readerSwitch.setTextOff(getString(R.string.disconnect));
40                        readerSwitch.setChecked(false);
41                        break;
42                   case SERVICE_STATE_INITIALIZING:
43                        readerSwitch.setTextOn(getString(R.string.initializing));
44                        readerSwitch.setChecked(true);
45                        break;
46                   case SERVICE_STATE_LISTENING:
47                        readerSwitch.setTextOn(getString(R.string.rfid_listening));
48                        readerSwitch.setChecked(true);
49                        break;
50                   case SERVICE_STATE_ENDPROCESS:
51                        readerSwitch.setTextOff(getString(R.string.end_process));
52                        readerSwitch.setChecked(true);
53                        break;
54                   default:
55                        break;
56              }
57         }
58    }
59
```

ソースコード 4.2 MainFragment クラス.

4.2.4. PrefsFragment

本アプリに設定を行うのがこの Fragment の役割である．Android では設定画面をつくるための Preference Fragment というものが用意されているため，これを継承し，拡張して作成している．図 4.8 の左側がこの PrafsFragment で，右側の 4 つの画面が，各項目を選択した場合の画面遷移である．この作り方をする場合，画面のビューはほぼ XML のみで定義することができる（ソースコード 4.3）．しかし，この手法でできることは非常に限られているため，一部のメニューで拡張を行っている．

- Bluetooth 設定：Android の Bluetooth 設定画面に飛び，リーダとのペアリングをユーザに行ってもらう
- リーダ選択：DeviceListFragment に飛び，接続対象のリーダをユーザに選択してもらう
- 管理対象登録モード：モードの On/Off を SharedPreference に保存する
- 推測演算時刻指定：TimePickerPreference を使い，1 日 1 回しなければならない推測演算を行う時間を指定してもらう

順番に説明する．まず，「Bluetooth 設定」については，Android における Bluetooth のペアリングを個々のアプリで行わないというルールを遵守するためのものである．このルールに則り，OS 側の Bluetooth 設定画面を呼び出すこととした．ここでペアリングを行ったのち，再び本アプリで設定を行ってもらう．

続いて「リーダ選択」は，ユーザにペアリング済み Bluetooth 機器のリストを提示し，どれがリーダであるかを選択してもらうものだ．リーダを 1 台しか持っていないユーザの場合あまり意味を成さないが，家族で複数台持っているようなユーザを想定し，このような画面を用意した．

「管理対象登録モード」については 4.2.7 節で詳しく議論するが，物-事象関係性推測演算の対象となる RFID を登録するためのモードである．

最後に「推測演算時刻指定」は，物-事象関係性推測演算を行う時間を推測するものである． 4.2.3 節でも触れたように，PresumeService は 1 日 1 回，指定時刻に立ち上がる．この時間を指定するためのものである．なお，この TimePickerPreference 画面については，理音伊織氏が Apache License 2.0 下で提供を行っているもの[55]を利用させていただいた．

図 4.8　PrefsFragment における画面遷移.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

    <PreferenceCategory android:title="@string/general_setting">
         <EditTextPreference android:summary="@string/summary_username"
                   android:title="@string/username"/>
    </PreferenceCategory>

    <PreferenceCategory android:title="@string/system_setting">
         <PreferenceScreen android:summary="@string/summary_goto_bluetooth_setting"
                   android:title="@string/goto_bluetooth_setting"
                   android:key="bluetooth_setting"
                   android:action="android.provider.Settings.ACTION_BLUETOOTH_SETTINGS"/>
         <PreferenceScreen android:summary="@string/summary_select_reader"
                   android:title="@string/select_reader"
                   android:key="device_select"/>
         <SwitchPreference android:summary="@string/summary_management_tag_collect"
                   android:title="@string/management_tag_collect"
                   android:key="management_tag_collect"
                   android:switchTextOff="OFF"
                   android:switchTextOn="ON"/>
         <jp.ac.uec.is.spa.takamaru.wasuremono.TimePickerPreference
                   android:summary="@string/summary_puresume_time"
                   android:title="@string/puresume_time"
                   android:dialogTitle="@string/puresume_time_dialog"
                   android:dialogMessage="@string/puresume_time_dialog_message"
                   android:key="presume_time"
                   defaultHour="2"
                   defaultMinute="0"
                   is24Hour="false"  />
    </PreferenceCategory>
</PreferenceScreen>
```

ソースコード 4.3　PrefsFragment の画面を生成する XML.

4.2.5. DeviceListFragment

PrefsFragment でリーダ選択という項目をタップすると表示されるのが，この DeviceListFragment である．この Fragment ではリスト表示を採用しており (図 4.9)，Android から取得したペアリング済みの機器を並べて表示している．ここで選択された機器の MAC アドレスが SharedPreference に保存され，ReaderService にて利用される．

この MAC アドレスを保存する部分の処理をソースコード 4.4 に示す．

図 4.9 リーダを選択する画面.

```
1  //        ペアリング済みデバイスのどれかを選択した場合の処理
2          listView1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
3               @Override
4               public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
5                    Map<String, String> map =
                          (Map< java.lang.String,String>)parent.getItemAtPosition(position);
6                    String device_name = map.get("device_name");
7                    String device_mac = map.get("device_mac");
8                    Log.d(TAG_FOR_LOG, String.format("BTdevice onClick name:%s, device_mac:%s",
                          device_name, device_mac));
9                    SharedPreferences sharedPreferences =
                          PreferenceManager.getDefaultSharedPreferences(getActivity());
10                   SharedPreferences.Editor pref_edit = sharedPreferences.edit();
11                   pref_edit.putString("Reader_MACaddress", device_mac);
12                   pref_edit.commit();
13                   ((MainActivity)getActivity()).
                          changeFragment(MainActivity.PREFERENCE_FRAGMENT);
14              }
15         });
```

ソースコード 4.4 DeviceListFragment において接続するリーダを選択した時の処理.

4.2.6. CommonDialogFragment

4.2.2 節で論じたように，本アプリにおいては共通のダイアログ用 Fragment を用意した．ユーザに注意喚起を行う際は，タイトル，メッセージ， OK/Cancel ボタンの有無, ボタン押下時の動作を指定する変数と共にこの Fragment を呼び出せば良い．（4.2.4 節で論じた TimePickerPreference のダイアログは例外）

この Fragment に関する実装をソースコード 4.5 とソースコード 4.6 に示した．ボタン押下時の動作切り分けは，呼び出されるメソッドが抽象メソッドになっているため，呼び出し元でソースコード 4.5 のように指定することができる．今回掲載した MainActivity 上では，OK ボタンを推した場合の処理を 3 種類規定しており，ただダイアログを閉じる場合と Activity を停止状態にする場合（finish()），アプリ全体を終了する場合（moveTaskToBack(true)）がある．また，キャンセルボタンについては常にただダイアログを閉じるようになっている．

```
1     /**
2      * AlertDialog において OK ボタンを押した時の処理
3      * @param positiveButtonType OK ボタンの処理パターン
4      *              DIALOG_POSITIVE_CLOSE, DIALOG_POSITIVE_FINISH, DIALOG_POSITIVE_MOVETASKTOBACK
5      */
     public void doPositiveClick(int positiveButtonType) {
6         switch (positiveButtonType){
7             case(DIALOG_POSITIVE_CLOSE):
8                 break;
              case(DIALOG_POSITIVE_FINISH):
9                 finish();
                  break;
10            case(DIALOG_POSITIVE_MOVETASKTOBACK):
11                moveTaskToBack(true);
12                break;
13            default:
                  finish();
14        }
15    }
16    /**
17     * AlertDialog で NG ボタンを押した時の処理
18     */
19    @Override
20    public void doNegativeClick() {
21        //ただ閉じるだけ
22    }
```

ソースコード 4.5　MainActivity 上で指定された OK/Cancel ボタンの処理．

```
1       /**
2        * AlertDialogの生成
3        */
4       @Override
5       public Dialog onCreateDialog(Bundle savedInstanceState) {
6            Log.d(TAG_FOR_LOG, "onCreateDialog");
7            String title = getArguments().getString("title");
8            String message = getArguments().getString("message");
9            int type = getArguments().getInt("type");
10           final int positiveButtonType = getArguments().getInt("positiveButtonType");
11
12           AlertDialog.Builder alert = new AlertDialog.Builder(getActivity())
13                     .setTitle(title)
14                     .setMessage(message)
15                     .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
16                          @Override
17                          public void onClick(DialogInterface dialog, int which) {
18 //                              OKボタンが押された時
19                               Log.d(TAG_FOR_LOG,"onClick OK Button");
20                               listener.doPositiveClick(positiveButtonType);
21                               dismiss();
22                          }
23                     });
24           if (type == MainActivity.DIALOG_TYPE_OK_CANCEL) {
25 //             NGボタンも付ける場合
26                alert.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
27                     @Override
28                     public void onClick(DialogInterface dialog, int which) {
29 //                       Cancelボタンが押された時
30                          Log.d(TAG_FOR_LOG,"onClick Cancel Button");
31                          listener.doNegativeClick();
32                          dismiss();
33                     }
34                });
35           }
36           return alert.create();
37      }
```

ソースコード 4.6　CommonDialogFragmentにおける処理の一部.

4.2.7. ReaderService

本アプリケーションにおいて中心となるのがこの ReaderService である．Android において，Service とはバックグラウンド動作するものであり，基本的には明確に終了処理を行わない限り動き続けるものである．そのため，今回のアプリでは図 4.3 で示した常駐処理を司る．また，ユーザへの忘れ物通知とユーザからフィードバック処理も，この ReaderService が発行している．

この Service には 2 つのモードがある．Default モードと Management モードである．Default モードは通常のセンシングを行い，忘れ物が発見されればアラートを出すモードだが，Management モードでは RF タグを発見するとその ID を SharedPreference に書き込む．管理対象の物を登録するためのモードなのである．4.2.4 節で管理対象登録モードと論じたのが, この Management モードのことである．提案システムでは，持ち歩かない物のタグがセンシングに含まれると不必要な演算や誤ったアラートが出てしまうことに繋がる．そこで予めシステムに管理対象の RFID を指示する必要があるが, ユーザが 1 つ 1 つ入力するのは非常に手間である. そこで, Management モードでこの ReaderService を立ち上げると, 周辺にある物の RFID を自動的に管理対象として登録してくれるのである．その性質から，基本的にはアプリ利用開始時に 1 回利用すれば良いモードであり，一旦登録を終えればこのモードを使う必要は基本的にない．持ち歩く物が増えた場合にのみ，4.2.4 節で説明した設定画面において管理対象登録モードを ON にして, 再びこのモードで ReaderService を立ち上げれば良い．

具体的に処理の流れを説明するにあたり, まず両モード共通の点について議論する．

この Service は，4.2.1 節で説明したリーダの SDK を使った処理が中心となる．Service も Activity 同様のライフサイクルが定められているが, Service のそれは非常に単純で，onCreate()→onStartCommand()→（様々な処理）→onDestroy()という一本道だけである．また，常駐するのでループで処理を継続させたりする必要はない．

具体的には，図 4.4 のフローのうち,「setOnDotrEventListener()メソッドを実行」「connect()メソッドでリーダに接続」「リーダに対して各種設定を行う」「inventoryTag()メソッドで読み取り開始」までを onStartCommand()のなかで行っている．なお，接続先については，DeviceListFragment で指定された MAC アドレスを SharedPreference から読み出している．また，平行して，フローのどこまで進んでいるかを Broadcast している（この Broadcast を受け取っているのは，4.2.2 節で説明した MainActivity である）．inventoryTag()メソッドを実行したあとは，4.2.1 節で説明したように，RF タグが発見されると onInventoryEPC イベントが発生するが，この後の処理はモードによって異なる．

・Default モードの場合

Default モードの場合，onCreate()における処理が 1 つ増える．それが tagTimeoutCount スレッドの立ち上げである．3.1 節で議論したたように，RFID データストリームは信頼性が低い．読み落としは頻繁におこる．そこでサイズを固定したウィンドウを適用し，抜け落ちたデータを適宜埋めて連続性のあるデータに整える必要がある．この，整える作業を行うのが tagTimeoutCount スレッドの役割である．

このスレッドに関連する処理の一部をソースコード 4.7 に示した．ここでやっているのは，読み込まれた各タグについて[tagID，startTime，windowStartTime，EndTime]というセットを作成する作業である．あるタグが初めて読み込まれると，その読み込まれた時間が startTime と windowStartTime に書き込まれる(19 行目〜)．そして，再び読み込まれた場合には，現在時刻で windowStartTime のみ更新する (27 行目〜)．そして，windowStartTime から既定の window サイズぶんの時間が経過すると(35 行目で判定)，その時の時刻を EndTime として [tagID, startTime, endTime] というセットでデータベースに書き込みを行う（36 行目，57 行目〜）．これにより，ストリーミングデータの穴埋めをすると同時に，点だったデータを繋いで時間的継続性のあるデータに変換することができる．

データベースへの保存については， Apache License 2.0 のもとで提供されている Active Android[56]という Android 向け O/R Mapper を利用した．これは，ソースコード 4.8 のようなクラスを準備すると，ソースコード 4.7 の 63〜67 行目のような形で非常に簡単に SQLite の書き込みができるものである．読み込みや上書き，検索についても同様に簡単な記述で扱えるため，今回は全面的にこれを利用している．

ソースコード 4.7 の 21 行目と 38 行目に checkBrings()というメソッドがあるがこれは，ユーザに通知を出すためのメソッドである．通知に関連する処理の一部をソースコード 4.9 に示した．

ソースコード 4.9 の checkBrings()メソッド（5 行目〜）では，SQL 文でその物が必要物としてデータベースの TodayBrings テーブルに保存されているか確認する（6 行目〜）．もしテーブル中に存在すれば，持っていかなければならない物であるから，通知を表示/消去する．なお，玄関タグを読み込む＝外出・帰宅する際には，必要物リスト中の物を全て持っているか確認する goOutNotification()メソッドが呼ばれ，必要な通知を行う．

実際に通知を出しているのは，sendNotification()メソッドである（22 行目〜）．基本的な流れは NotificationBuilder を作成（35 行目〜）した後，NotificationManager を取得（55 行目〜），取得された Manager に対して Notification につける ID と共に Builder を渡すことで通知が発行される．

今回，4.1.2 節で議論したように，ユーザに忘れ物を通知する手段として Android

Wear を使用するが， Android Wear で画面表示を行うには 2 つの手段がある．スマートフォン側で通知を表示するとそれが Android Wear 側にも同期されることを利用する方法と，Android Wear アプリを作成し，それとスマートフォン側のアプリで通信を行うことで画面表示を行う方法である．今回は，より簡易な手法として前者の方法を用いている．

Android Wear への通知送信は Android 側で勝手にやってくれるため，開発者は特に何もしなくても通知を送ることができるが，今回は 2 つのカスタマイズを行った．具体的には，Android Wear の画面上に 1 つボタンを追加し，「通知が出ているが忘れ物はない＝推測は間違えている」ということをシステムに対してフィードバックがかけられるようにした（図 4.10）．左の画面で背景色が橙色になっているが，これは 3.2.4 節で説明した通知レベル（推測の信頼度）に応じて色が変わる．High であれば赤色，Middle であれば橙色，Low であれば水色になる．

これらのカスタマイズも NotificationBuilder にセットする必要がある．まず，31 行目〜で Android Wear 用の Action を用意する．この Action が先に述べたボタンのことであり，ボタンの画像を指定すると共に，ボタンが押された場合にアプリに戻ってくる Intent をセットする（33 行目）．この Action を Builder にセットする（51 行目）ことで，通知自体は Android スマートフォンと Android Wear の両方に出るものの，Android Wear では追加のボタンが表示されるようになる．

背景色の変更については少し事情が異なり，Android スマートフォンと Android Wear 共通で使われる画像ファイルを指定することで行う（27 行目〜，43 行目）．そもそも，Android Wear で表示される通知の背景は Android スマートフォンの通知領域（図 4.11）で使われているアイコンを表示するものであり，Android Wear だけ特別のものを使うということを考慮されていない．そこで，この共通のアイコン画像を通知レベルに応じて切り替える（27 行目〜）ことで，通知の背景色を変えることとした．

以上の処理は全て家の外でのみ処理して欲しい事柄である．自宅で収集したデータは持ち物の推測演算には使えないし，通知が出ても困る．そこで，3.3 節で議論したとおり，玄関に RF タグを 1 枚貼付し，これを読み取ったら自宅→外出先，もしくは外出先→自宅とモード切り替えを行うこととした．

```
1     /**
2      * RFID の低信頼性に対応するためのスレッドを動かし始めるためのメソッド
3      */
4     public void tagTimeoutCount() {
5          new Thread(new Runnable() {
6               @Override
7               public void run() {
8                    mRunningThread = true;
9                    while (mRunningThread) {
10                        endCheck();
11                   }
12              }
13         }).start();
14    }
15
16    /**
17     * 新しいタグが読み込まれた場合に利用する
18     */
19    public void writeStartTime(String tagId) {
20         startTimes.put(tagId, System.currentTimeMillis());
21         checkBring(tagId);
22    }
23
24    /**
25     * 継続してタグが読み込まれた場合に利用する
26     */
27    public void writeWindowStartTime(String tagId) {
28         windowStartTimes.put(tagId, System.currentTimeMillis());
29    }
30
31    public void endCheck() {
32         Long endTime = System.currentTimeMillis();
33         for (String tagId : windowStartTimes.keySet()) {
34              Long windowStartTime = windowStartTimes.get(tagId);
35              if (endTime - windowStartTime >= MainActivity.WINDOW_SIZE) {
36                   saveDatabase(tagId, startTimes.get(tagId), endTime);
37   //              忘れ物アラートを出す
38                   checkBring(tagId);
39                   removeList.add(tagId);
40              }
41         }
42   //    不要なものを Map から削除する処理
43         for (String key : removeList) {
44              windowStartTimes.remove(key);
45              startTimes.remove(key);
46         }
47         removeList.clear();
48
49    }
50
51    /**
52     * データベースへの保存を行う
53     * @param tagId      タグの ID
54     * @param startTime   タグが見つかり始めた時間
55     * @param endTime     タグが消えた時間
56     */
57    public void saveDatabase(String tagId, long startTime, long endTime) {
58         Log.d("save database", "tagID=" + tagId);
59         Log.d("save database", "    startTime=" + startTime + " endTime=" + endTime);
60         Log.d("save database", "    Duration" + (startTime - endTime));
61
62   //    データベースへの書き込み処理をする
63         HavedItemsLog havedItemsLog = new HavedItemsLog();
64         havedItemsLog.tagId = tagId;
65         havedItemsLog.startTime = startTime;
66         havedItemsLog.endTime = endTime;
67         havedItemsLog.save();
68    }
```

ソースコード 4.7　TagTimeoutCount スレッドに関連する処理.

```
1  package jp.ac.uec.is.spa.takamaru.wasuremono;
2
3  import com.activeandroid.Model;
4  import com.activeandroid.annotation.Column;
5  import com.activeandroid.annotation.Table;
6
7  @Table(name = "havedItemsLog")
8  public class HavedItemsLog extends Model{
9
10     @Column(name = "tagId")
11     public String tagId;
12
13     @Column(name = "startTime")
14     public long startTime;
15
16     @Column(name = "endTime")
17     public long endTime;
18
19     public HavedItemsLog(){
20          super();
21     }
22
23     public HavedItemsLog(String tagId, long startTime, long endTime){
24          super();
25          this.tagId=tagId;
26          this.startTime = startTime;
27          this.endTime = endTime;
28     }
29 }
```

ソースコード 4.8 ActiveAndroid を利用するために必要な TagStore.java ファイル.

```
1       /**
2        * タグが出現・消滅した際に，通知を出す必要がある物（＝持ち物リストに含まれるか）か確認、通知を出す/消す
3        * @param tagId
4        */
5       public void checkBring(String tagId) {
6            List<TodayBrings> eventList = new Select()
7                    .from(TodayBrings.class)
8                    .where("day = ? and tagId = ?", DateFormat.format
                              (MainActivity.DATE_PATTERN, System.currentTimeMillis()), tagId)
9                    .orderBy("Name ASC")
10                   .execute();
11           if (eventList.size() == 0) return;   //eventList が 0 ということは必要な持ち物ではない
12           if (startTimes.containsKey(tagId)) {      //今持っている物であるかをチェック
13                deleteNotification(tagId);
14           } else {
15                sendNotification(tagId);
16           }
17      }
18      /**
19       * 通知を出すメソッド　予め必要な持ち物であるかはチェックしておくこと
20       * @param tagId   通知を出したいタグの ID
21       */
22      private void sendNotification(String tagId) {
23           Intent intent = new Intent(this, MainActivity.class);
24           PendingIntent contentIntent = PendingIntent.getActivity(this, 0, intent, 0);
25 //        持ち物 DB から関係事象・信頼度を読み出し，Notification を開いたときのサブタイトル部分を纏める
26 //         （中略）
27 //        LargeIcon の Bitmap を生成　信頼度に応じて選択する
28           Bitmap largeIcon;
29           if (notificationLevel == MainActivity.HIGH_NOTIFICATION){
                  largeIcon = BitmapFactory.decodeResource(getResources(), R.drawable.red_icon);
30 //         （中略）
31 //        Android wear 専用の Action を追加する
32           Intent wearIntent = new Intent(getApplicationContext(), MainActivity.class);
33           PendingIntent notWasuremonoIntent = PendingIntent.getActivity(this,0,wearIntent, 0);
34           NotificationCompat.Action action = new NotificationCompat.Action
                        (R.drawable. proofreading_icon,getString(R.string.not_wasuremono),
                        notWasuremonoIntent);
35 //        NotificationBuilder を作成
36           NotificationCompat.Builder builder
                        = new NotificationCompat.Builder(getApplicationContext())
37                   .setContentIntent(contentIntent)
38 //                 ステータスバーに表示されるテキスト
39                   .setTicker(getString(R.string.notification_ticker))
40                   .setSmallIcon(R.drawable.ic_launcher)
41 //                 Notification を開いたときに表示されるタイトル
42                   .setContentTitle(getString(R.string.notification_context_title))
43 //                 Notification を開いたときに表示される&Wear の背景になるアイコン
44                   .setLargeIcon(largeIcon)
45 //                 カードのグループを設定する（Android Wear 用）
46                   .setGroup(MainActivity.NOTIFICATION_GROUP)
47 //                 通知するタイミング
48                   .setWhen(System.currentTimeMillis())
49 //                 通知時の音・バイブ・ライト
                     .setDefaults(Notification.DEFAULT_SOUND
                                | Notification.DEFAULT_VIBRATE
                                | Notification.DEFAULT_LIGHTS)
50 //                 Wear 用のアクションをセットする
51                   .extend(new NotificationCompat.WearableExtender().addAction(action))
52 //                 スマホ上で BigStyle になるようにサブタイトル部分を設定
53                   .setStyle(new NotificationCompat.BigTextStyle()
                                .bigText(contextText.toString()));
54           // NotificationManager を取得
55           NotificationManager manager
                        = (NotificationManager) getSystemService(Service.NOTIFICATION_SERVICE);
56           // Notification を作成する
57           manager.notify(convertTagIdToNotifyId(tagId), builder.build());
58      }
```

ソースコード 4.9　忘れ物の確認・通知に関する処理の一部.

図 4.10　Android Wear 上におけるフィードバック機構.

図 4.11　Android スマートフォン上での通知.

・Management モードの場合

Management モードの場合は，もっと単純である．onInventoryEPC()イベントを受け取ったら，その RF タグが既に管理対象 RFID リストに入っているかを確認する．入っていればこれを無視し，入っていなければこれをリストに追加する．

基本的には，Default モード・Management モード共に，センシングが開始されればずっと処理が続く．しかし，途中でリーダとの接続が切れてしまったり，そもそも最初にリーダと接続できなかったり，あるいは MainFragment 上にあるリーダ接続ボタンがユーザによって Off にされたりした場合には，この Service を終了するための onDestroy()が呼ばれる．onDestroy()では，リーダとの接続終了処理を行うほか，Default モードの場合には読み込みが続いていたタグについて現在時刻を endTime としてセットし，ソースコード 4.7 の saveDatabase()メソッドでデータベースに保存する．

4.2.8. PresumeService

PresumeService は，毎日定められた時間に起動し，物–事象関係性推測アルゴリズムを実行する部分である．定時に起動することについては，4.2.2 節で論じたように AlarmManager という Android の提供する機能を利用している．このため，ReaderService とは異なり，常駐し続ける必要がなく，計算が終わり次第 stopSelf(); を呼んで自らプロセスを終了する．なお，起動時刻については，4.2.4 節で議論したように，PrefsFragment のメニュー「推測演算時刻指定」にて指定が可能である．

この Service における最も重要な役割は，ReaderService で収集されたセンシングデータを元に，3.2.3 節の式(3.4)を用いて，表 3.3 のようなテーブルをアップデートすることである．この実装の一部をソースコード 4.10 で示す．TagEventRelationship という型があるが，これは ActiveAndroid の使用に際して作成したモデルクラスである（構造については，4.2.10 節で説明する）．

計算の主体は dailyUpdate()メソッド（1 行目〜）で，これに対し配列に格納された事象（もしくは予定タグ）と持ち物の RFID を与えることで，DB の更新までがなされる．個々の物や事象が今日存在するかについては，それぞれ rfidMap と eventMap という HashMap で管理しており，まず preparationHashMap()メソッドがデータベース上に存在する全ての物と事象（もしくは予定タグ）について HashMap に「ない」として登録する．続いて今日存在した物と事象については，「ある」として HashMap を上書きする（5 行目〜）．ここまでで，今回演算に加える日を含む全期間において存在した物と事象が HashMap 上に登録されていることになる．

得られた物と事象のリストを用いて，物と事象全ての組み合わせについて物–事象関係性推測演算を行う（12 行目〜）．この際，まず readPresume()メソッド（45 行目〜）を使って今回計算を行う組み合わせについて既に DB に保存されているかどうかを確認し，保存されていればその TagEventRelationship 型のデータの取得を（60 行目），存在していなければ新たに TagEventRelationship 型のデータを用意する（54

行目〜）．updatePresume()メソッド（28 行目〜）に得られた TagEventRelationship 型データと計算対象の物と事象のあるなしを引数で与えると，式(3.4)に相当する計算を行い (32 行目〜), 更新された TagEventRelationship 型データが戻ってくるので，これをデータベースに上書き保存する（16 行目）．

以上によって物–事象関係性推測の更新が完了すると，翌日の必要物リスト生成が可能となる．但し，MainFragment 上にある，忘れ物アラートスイッチが On になっている場合にのみリストの生成を行う．

まず翌日に存在する事象を全て取得する必要があるが, 今回は Android 標準のカレンダーアプリから取得した予定と，Livedoor Weather Hacks[52]から取得できる天気予報についてのみ実装を行った．得られた事象のうち，予定については予定タグの分離を行った上で，それぞれをデータベースの TagEventRelationship テーブルに問い合わせを行い，それぞれの$P_t(Y|X)$に応じてリストに追加するかを判断する．この判断基準については 3.2.4 節で議論したとおりである．

```
1     public void dailyUpdate(String[] event, String[] rfid) {
2  //       eventMap と rfidMap を準備，現在 DB 上にある全ての事象と物が HashMap 上に存在するようにする
3           preparationHashMap();
4  //       今日存在した事象や物について，ハッシュマップ上で「あった」と登録
5           Integer in = new Integer(MainActivity.IS_THERE);
6           for (int i = 0; i < event.length; i++) {
7                eventMap.put(event[i], in);
8           }
9           for (int i = 0; i < rfid.length; i++) {
10               rfidMap.put(rfid[i], in);
11          }
12          for (Map.Entry<String, Integer> targetRfid : rfidMap.entrySet()) {
13               for (Map.Entry<String, Integer> targetEvent : eventMap.entrySet()) {
14                    TagEventRelationship beforePresume
                                   = readPresume(targetRfid.getKey(), targetEvent.getKey());
15                    TagEventRelationship newPresume
                                   = updatePresume(beforePresume, targetEvent.getValue().intValue(),
                                        targetRfid.getValue().intValue());
16                    newPresume.save();
17               }
18          }
19     }
20     /**
21      * 1 つの物・事象（or 予定タグ）セットについて，1 日分のアップデートを行う
22      * 複数日纏めて処理するときは呼び出し元でループを回すこと
23      * @param presume      TagEventRelationship
24      * @param newDayEvent   その日にその事象があれば MainActivity.IS_THERE なければ IS_NOT_THERE
25      * @param newDayRfid   その日にその物があれば MainActivity.IS_THERE なければ IS_NOT_THERE
26      * @return            TagEventRelationship
27      */
28     public TagEventRelationship updatePresume
                    (TagEventRelationship presume, int newDayEvent, int newDayRfid){
29          double beforeNumerator = presume.numerator;
30          double beforeDenominator = presume.denominator;
31
32          double newNumerator
                         = beforeNumerator * MainActivity.PAST_IMPORTANCE + newDayEvent * newDayRfid;
33          double newDenominator = beforeDenominator * MainActivity.PAST_IMPORTANCE + newDayEvent;
34
35          presume.numerator = newNumerator;
36          presume.denominator = newDenominator;
37          return presume;
38     }
39     /**
40      * データベースからの読出しを行う
41      * @param tagId        タグの ID
42      * @param event        事象・予定タグ
43      * @return            TagEventRelationship
44      */
45     public TagEventRelationship readPresume(String tagId, String event) {
46          List<TagEventRelationship> beforePresumes = new Select()
47                         .from(TagEventRelationship.class)
48                         .where("tagId = ? and event = ?", tagId, event)
49                         .execute();          //データベース読み込み
50          int size = beforePresumes.size();
51          if(size>1) return null;
52          TagEventRelationship returnPresume;
53          if (beforePresumes.size() == 0) {         //この組み合わせが初出だったら
54               returnPresume = new TagEventRelationship();
55               returnPresume.tagId = tagId;
56               returnPresume.event = event;
57               returnPresume.denominator = 0;    //分母
58               returnPresume.numerator = 0;    //分子
59          } else {
60                         returnPresume = beforePresumes.get(0);
61          }
62          return returnPresume;
63     }
```

ソースコード 4.10　PresumeService における物–事象関係性推測演算の一部.

4.2.9. havedItemsLog

havedItemsLog は，リーダによって収集された RFID のログが保管されるデータベーステーブルである．このテーブルの要素のデータ構造は表 4.4 のとおりである．なお，Id カラムについては，Activeandroid が自動的に生成し，主キーとなっている．

なお，UNIX 時間とは 1970 年 1 月 1 日午前 0 時 0 分 0 秒からの経過ミリ秒数である．通常は秒単位だが，Android ではミリ秒単位を標準としている．

表 4.4 havedItemsLog テーブルの形式.

Id	tagId	startTime	endTime
int （自動生成）	String	long (UNIX 時間)	long (UNIX 時間)

4.2.10. tagEventRelationship

tagEventRelationShip は，物–事象関係性推測演算の結果が保管されるデータベーステーブルである．このテーブルの要素のデータ構造は表 4.5 のとおりである．

表 4.5 tagEventRelationship テーブルの形式.

Id	tagId	event	denominator	numerator
int （自動生成）	String	String	double	double

4.2.11. todayBrings

todayBrings は，直近 1 日分の持ち物が保管されるデータベーステーブルである．このテーブルの要素のデータ構造は表 4.6 のとおりである．

表 4.6 tagEventRelationShip テーブルの形式.

Id	day	event	tagId	notification Level
int （自動生成）	long (UNIX 時間)	String	int	int

4.3. 実装したシステム利用の流れ

この節では，4.2 節で実装したアプリを実際に利用するための流れを説明する．

4.3.1. 事前準備

本アプリを用い, 忘れ物通知を受け取るためには, 以下のような準備が必要である．図 4.12 はこの流れを示したものである．なお，既に Android スマートフォンと Android Wear 端末はペアリングされ，利用できる状態になっていることを前提とする．

- 持ち歩く可能性のある物全てに RF タグを貼付する
- 玄関に専用の RF タグ（玄関タグ）を貼付する
- リーダと利用する Android スマートフォンをペアリングする
- 設定画面の玄関タグ登録メニューで，玄関に貼付した RF タグの ID を入力する
- 設定画面のリーダ選択メニューから，接続するリーダを選ぶ
- 管理対象登録モードでリーダ接続を行い，RF タグを貼付した物にかざす
- 忘れ物アラートは Off のまま，しばらく持ち物のログを収集する

持ち物のログを収集する期間については特に制限を設けていないが，ログが少なければ少ないほど，誤った通知が出てしまう可能性が高まる．ログ収集期間と推測制度に関しては，5.2.3 節にて評価実験を行っている．

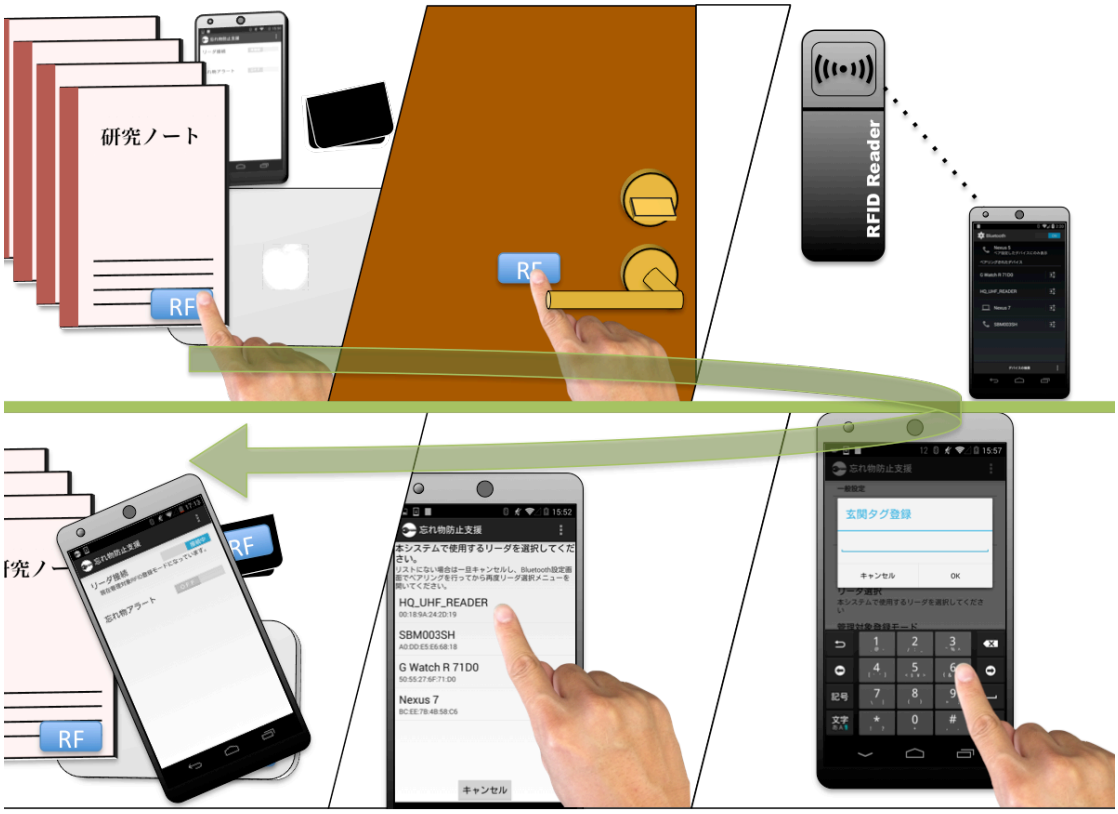


図 4.12　システム利用準備のイメージ.

4.3.2.　日常における利用の流れ

前節で論じたような準備を行うことで，実際に忘れ物をした時にシステムから通知してもらえるようになる.忘れ物アラートを ON にして 1 度リーダ接続を開始すれば，特別な操作をユーザがする必要はない（図 4.13）．いままでどおりにカレンダーに予定を登録していれば，自動的にアラートが提示されるようになる．また，アラートが間違っていた場合には，Android Wear 上で「忘れ物はありません」というボタンをタップすることで，物–事象関係性推測演算の結果にフィードバックを与えることができる（図 4.10）.

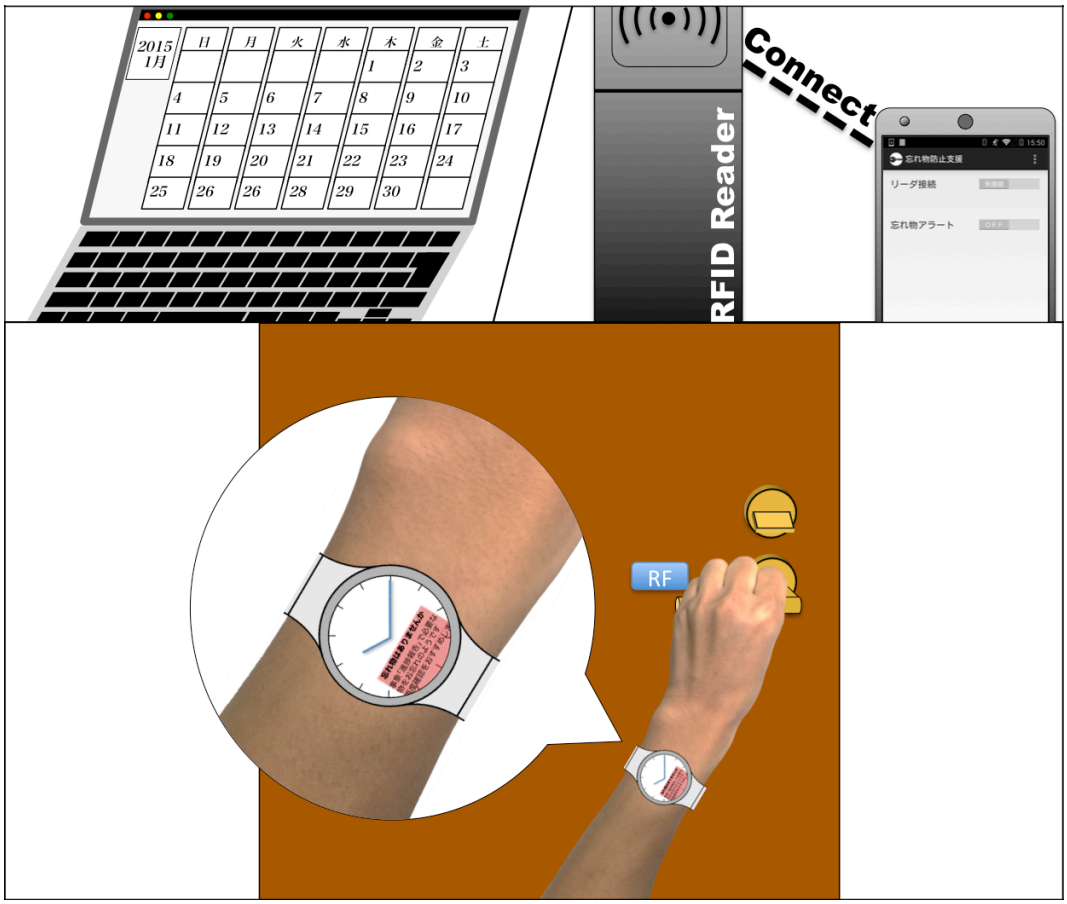


図 4.13　日々の利用イメージ.

第5章 評価実験

今回，提案した手法の有効性を確認するにあたり，大きく２つの実験にわけて評価を行った．連結テストを行うためには１回分のデータ収集だけで数ヶ月かかることが見込まれるため，分離したものである．

- 現実世界において，提案した RFID データ収集法で有効に持ち物を認識することが可能か
- 持ち物と事象のセットがシステムに対して与えられたとき，有効に物–事象関係性推測演算が動作し，持ち物を推測できるか

前者は 5.1 節で，後者は 5.1.3.1 節で実験の設定と結果，その考察を論じる．

5.1. RFID データ収集実験

現実世界において提案した RFID データ収集法で有効に持ち物を認識することが可能であるかを確認するため，今回２段階にわけて実験を行うこととした．まず，提案した手法においてウィンドウのサイズを決定するための実験を行い，その後決定したウィンドウのサイズを用いて実装されたシステムによって持ち物が正しく掌握できているか実験する，

なお，リーダは電波を照射する際に照射と休止を繰り返すが，今回はその照射時間を 200ms として全ての実験を行った．

5.1.1. 使用した RF タグ

本実験においては，２種類の RF タグを用意して実験を行った．１つめが，富士通フロンテック社と大日本印刷社が共同開発した書類管理用ラベルタグ TFU-TL4AxB[19]である．もう１点は Omni-ID 社の Omni-ID MAX Label（015-US）[57]である．

これらの RF タグの写真はそれぞれ図 5.1，図 5.2 に示した．また，スペックシートは表 5.1 のとおりである．

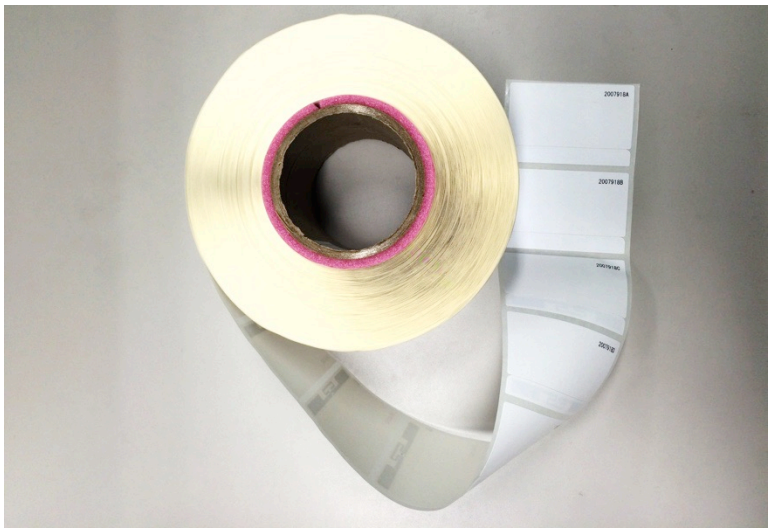


図 5.1　富士通フロンテック社と大日本印刷社が共同開発した書類管理用ラベルタグ.

図 5.2　Omni-ID 社が開発した Omni-ID MAX Label.

表 5.1　書類管理用ラベルタグ及び Omni-ID MAX Label のスペックシート.

項目	仕様	
製品名	書類管理用ラベルタグ	Omni-ID MAX Label
型番	TFU-TL4AxA	015-US
サイズ	W79mm×D11mm	W80mm×D30mm×H3.8mm
周波数帯	920MHz 帯	920MHz 帯 （902～928MHz）
通信距離	6～7m	～12m
特徴	1,2mm 厚の書類などに添付し重ねても読める	金属への貼り付けに対応

5.1.2.　ウィンドウサイズの検討

3.1 節で論じたように，本研究では読み落としの多い RFID センシングデータに対し，固定サイズのウィンドウを時間的にスライドさせることによって穴埋めを行うこととしている．このウィンドウのサイズを決定するため，実際に持ち物と RFID リーダを持って大学と自宅間を往復し，RFID センシングデータを取得した．

ただ，第 4 章で実装したシステムでは，生データを保存する機能が存在しない．そこで，今回 4.1.1 節で示した RFID リーダに付属していたサンプルアプリ（スマートフォンとリーダで通信を行い，センシングデータを画面に表示する機能のみ存在するもの）にログを出力させることで，生のセンシングデータを取得することとした．

5.1.2.1.　ウィンドウサイズを検討する実験の環境・設定

実験においては，3つの持ち物を用意した. カード入れ, 手帳, ノート PC(Macbook) である．カード入れは胸ポケットに，手帳と Macbook は手提げ鞄に入れている．また，RFID リーダについては付属のストラップで首から下げて持ち運んだ（図 5.3）．また，目安として，リーダ本体にも 1 枚 RF タグを貼付した．

まず，予備実験として富士通フロンテック社と大日本印刷社が共同開発した書類管理用ラベルタグを用い, 15 分ほど歩行及び自転車乗車して RFID データを収集する実験を行った．しかし，バッグ内に入れた手帳とノート PC は，数分に 1 回受信できる程度と，非常に電波受信状況が悪かった．この原因として考えられるのは， RFID リーダと RF タグはいずれも指向性があるものであるため，その方向が噛み合わなかった可能性と，金属の塊であるノート PC が通信を阻害した可能性である．

そこで，改めて Omni-ID 社が開発した Omni-ID MAX Label を使用して実験をやり直した. 今回は自宅〜大学間を 1 往復した. 実験を行ったのは以下の時間帯である．

・行き：8 時 17 分 56 秒〜9 時 18 分 03 秒（1 時間 0 分 7 秒）

・帰り：23 時 42 分 49 秒〜翌 1 時 1 分 5 秒（1 時間 18 分 16 秒）

また，移動区間内には列車に乗っている時間が約 20〜25 分ほど含まれており，それ以外に自転車移動が 5〜10 分と徒歩移動が 15 分弱，電車の待ち時間が含まれている．このほか，帰宅時はコンビニへ寄り道をした時間が含まれる．

行きの電車内は比較的混雑をしており，主観的には混雑率 150%程度であった．帰宅時は夜遅く，混雑は行きほどではなかったものの，経路中の列車 1 本が終電であったために混雑率は 50%程度だった．

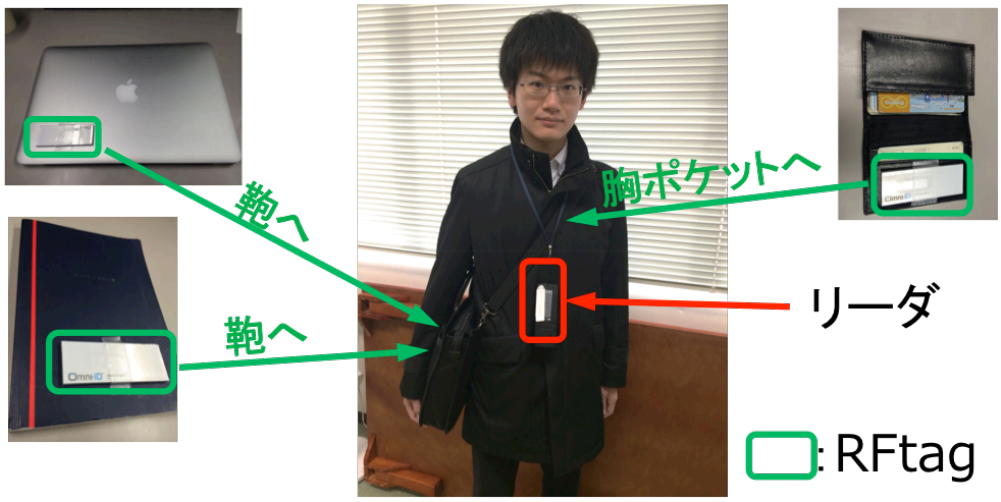


図 5.3　ウィンドウサイズ検討実験の様子.

5.1.2.2.　ウィンドウサイズを検討する実験の結果・考察

本実験はウィンドウサイズの検討を行うことが目的であるため，各 RF タグの受信時間間隔を分析した．その結果が，表 5.2 である．なお，今回は 1 秒以下については連続したデータだと判断している．

胸ポケットに入れたカードケースの RF タグは, RFID リーダからの距離が近いこともあり，読み落としは非常に少ない回数であった．行きに至ってはわずか 3 回だけであり，非常に良好な通信状態だったことが認められる．回数ベースでの分析においても，リーダ本体に基準として貼付した RF タグに比べ，往復合わせて 92.1%の回数の受信ログが存在した．

一方，手帳やノート PC については，物理的に離れていることに加え，リーダと RF タグが向かい合うことも難しく，データからは非常に苦しい通信状態が見て取れる．特に帰りは読み落としの継続時間が長くなる傾向が認められ，行きの手帳は最長でも 34 秒（電車に乗っていた時間帯）離れているのみであったが，帰りの手帳は 194 秒（自転車に乗っていた時間帯）も離れている例が存在した. 読み取り回数ベースでは，リーダ本体に貼付されたものの 18.1%であった．金属の塊であるノート PC はさらに悪い結果が出ており，行きは最長 228 秒（電車に乗っていた時間帯），帰りは 578 秒（自転車に乗っていた時間帯）と約 10 分間リーダに発見されない状態が続いていた．同様に読み取り回数ベースでは 44.3%である．

受信間隔がかなり開いてしまっている時間帯が散見され，あまり良い結果とは言い難いが，以上の結果を踏まえてウィンドウのサイズは 30 秒に設定することとした．

この設定であれば，行きと帰りの各 3 種類の RF タグ計 6 つのデータ全てで 75%以上の途切れを補正できることになるためである．ウィンドウをさらに大きくすれば，補正はさらに強くなりデータの途切れをさらに減らすことができるが，ユーザへの通知のタイミングが遅れることに繋がる．また，今後もう一つの「忘れ物」(置き忘れ）への拡張を考える際にも，大きすぎるウィンドウは問題である．そのため，今回はバランスを考えた値として，30 秒とすることにした．

表 5.2　リーダと物を持ち歩く実験において読み落としのあった回数.

	行き			帰り		
	カード入れ	手帳	ノート PC	カード入れ	手帳	ノート PC
1〜5 秒	3	21	151	26	19	25
6〜10 秒	0	1	30	2	6	6
11〜20 秒	0	2	18	2	4	10
21〜30 秒	0	0	7	0	4	3
31〜40 秒	0	1	4	0	1	1
41〜50 秒	0	0	2	0	2	2
51〜60 秒	0	0	1	0	2	1
61〜90 秒	0	0	7	0	1	2
91〜120 秒	0	0	0	0	1	2
121〜150 秒	0	0	1	0	1	0
151〜180 秒	0	0	0	0	1	1
181 秒〜	0	0	2	0	2	1

(単位：回)

5.1.3.　実装されたシステムによるセンシングデータ収集実験

5.1.2 節の検討により，ウィンドウのサイズを 30 秒とすることが決まった．これにより，第 4 章で実装されたシステムを実際に使用し，出力される時間的継続性のあるデータがどのようになっているか確認する実験を行った．

5.1.3.1.　実装されたシステムによるセンシングデータ収集実験の環境・設定

今回の実験においては，5 つの持ち物を用意した．カード入れ，財布，クリアフォルダに入れた書類，ノート PC（Macbook），ノート PC の充電器である．カード入れは胸ポケットに入れ，それ以外の物は全て肩掛け鞄に入れている．また，RFID リーダについては前の実験同様に，付属のストラップで首から下げて持ち運び，物に貼付する RF タグは Omni-ID 社の Omni-ID MAX Label を使用した．

今回も自宅〜大学間を 1 往復した．実験を行ったのは以下の時間帯である．

・行き：7 時 34 分 08 秒〜8 時 23 分 04 秒（0 時間 48 分 56 秒）
・帰り：20 時 50 分 49 秒〜21 時 34 分 05 秒（0 時間 43 分 16 秒）

行きの電車内は比較的混雑をしており，主観的には混雑率 150%程度であった．また, 帰宅時は経路中に使用する列車の一方は混雑率 130%程度, もう一方は混雑率 50%程度だった．なお，5.1.2 節の実験に比べて移動時間が短いが，これは列車の待ち時間に端を成すものであり，経路としては変わりはない．

5.1.3.2.　実装されたシステムによるセンシングデータ収集実験の結果

今回の実験によって出力させるデータは，時間的なデータになっているため，実際の移動時間に対してどれくらいの時間システムに「持っている」と認識されていたかを調べ，これを評価するものとした．

結果は表 5.3，表 5.4 のとおりである．財布が比較的低い値を示しているものの，全体的には 70〜80%という値が出た．

財布が低い値を示している原因を推定するのは困難であるが，硬貨に電波が干渉を受けた可能性や，鞄の中で PC と RF タグが密着して干渉を受けた可能性があると考えられる．この他，ノート PC が良い値を示していることが見て取れるが，ノート PC の RF タグは鞄の中でも中央〜上の方にあったため，鞄の底の方にあった財布や充電器の RF タグに干渉を受けにくかったことなどが考えられる．

表 5.3　実装されたシステムによるセンシングデータの収集結果（行き）.

	行き				
	カード入れ	財布	書類	ノート PC	充電器
「ある」と認識していた時間（秒）	2072	1716	2041	2484	2344
移動時間に対する割合（%）	70.57	58.45	69.52	84.6	79.84

表 5.4　実装されたシステムによるセンシングデータの収集結果（帰り）.

	帰り				
	カード入れ	財布	書類	ノート PC	充電器
「ある」と認識していた時間（秒）	1952	1658	1802	1986	1965
移動時間に対する割合（%）	75.19	63.87	69.41	76.5	75.69

5.1.3.3.　実装されたシステムによるセンシングデータ収集実験の考察

前項で論じたように，今回の実験では，Passive 型 RF タグとモバイル RFID リーダの組み合わせで持ち物を管理できるのは，全体の 70〜80%程度の時間であるということがわかった．この 70〜80%という値は理想的な値とは言い難く，現在の値では正しく持ち物を持っているにも関わらず忘れ物を知らせるアラートが出てしまうと考えられる．今現在持っている物に貼付された RF タグを正確に認識するということは，本研究の土台とも言うべきことであり，移動中に RF タグを貼付した物をモバイル RFID リーダで管理するためには，更なる改良や，運用上の工夫が必要であると言えよう．

例えば，確実に持ち物を把握したいときには RFID リーダを鞄に向けるようにしたり，鞄の中に RFID リーダをしまっておくことを検討したりといったことである．2.4 節で調べたように今後携帯電話と RFID リーダが一体化すれば，このような運用上の工夫はさらにしやすくなると考えられる．例えば，普段鞄の中に携帯電話を入れている人がいるが，このような人であれば今まで通りの使い方をするだけで，正確に持ち

物を管理することができると考えられる．

一方で，システム面の改良としては，人が歩いている時には持ち物が身の回りから消えることはないと仮定し，人の動作を加速度センサなどから推測してアラートを出すべき時・出す必要がない時を判断するアルゴリズムの導入などが考えられる．

この他，今後の技術革新によって RFID の電源変換効率や通信距離が更に向上すれば，提案通りの手法でほぼ問題なく稼働することも期待できる．このようなことを勘案すると，70〜80%という値は，Passive 型 RF タグとモバイル RFID リーダの組み合わせで持ち物を管理することについて，充分に可能性を示せたものと考える．

5.2. 必要な持ち物推定実験

続いて，物–事象関係性推測演算及び必要な持ち物の推定を正しく行うことができるかについて調べる．この実験は，

・必要な持ち物推定実験 1：パラメータkの調整
・必要な持ち物推定実験 2：データ蓄積期間と推測精度
・必要な持ち物推定実験 3：忘れ物率と推測精度

の 3 つにわかれている．

5.2.1. 必要な持ち物推定実験の設定

今回実装を行ったシステムでは，元々1 日 1 回 1 日分のデータを物–事象関係性推測演算に反映させる機能しかない．また，持ち物リストの生成も翌日分に限られる．そこで，第 4 章で実装したシステムに対し，事象とタグ ID のリストを入力すると事象収集部分をバイパスして物–事象関係性推測演算を行う機能を追加した．入力ファイルに記入がある日数分ループするようになっているため，この機能を使えば 100 日分の蓄積を一気に行ったりすることが可能になる．本節の実験は，この機能を利用して行うものとする．

また，長期間に渡る持ち物・事象のログデータを複数人分収集することは現実的に不可能であったため，今回は異なる特性を持ったユーザを複数人想定し，それらの特性に合う形で事象及び持ち物のログを生成した．

これらのユーザについて，定期的に存在する事象・予定タグを計 35 個用意した．以下のとおりである．

Daily，雨，晴，
【発表】，【会議】，【講義】，【試験】，【客先訪問】，【商品紹介】，【デモ】，【新規客先】，【バイト】，

講義 A，講義 B，講義 C，講義 D，講義 E，講義 F，講義 G，講義 H，ゼミ，部内会議，社内会議，出張，A 社打ち合わせ，B 社打ち合わせ，C 社打ち合わせ，D 社打ち合わせ，飲食店 A，百貨店 B，コンビニ C，飲み会，ライブ，説明会，買い物

このなかには 3 つ，特殊なものがある．まず「Daily」は，毎日必ず追加されるものである．例えば，財布や携帯電話など，日々の持ち物全てがこれに含まれる．また「雨」「晴」も特殊である．これらは相反する事象であり，決して同時に存在しない．しかし，どちらかは必ず存在することになる．今回，東京地区の 2013 年降雨・降雪日数が 98 日[58]であったことから，27%の確率で雨天になるように設定した．

また，【】がついているものは事象につく予定タグである．どの事象にこの予定タグが付くかは予め決定しており，

- 講義 A，講義 B，講義 C，講義 D，講義 E，講義 F，講義 G，講義 H
  - → 【講義】：必ず付加される
  - → 【試験】：ユーザ B〜D においてランダムに付加される
  - → 【発表】：ユーザ B〜D においてランダムに付加される
- ゼミ
  - → 【発表】：ユーザ B〜D においてランダムに付加される
- 部内会議，社内会議
  - → 【会議】：必ず付加される
  - → 【発表】：ユーザ B〜D においてランダムに付加される
- A 社打ち合わせ，B 社打ち合わせ，C 社打ち合わせ，D 社打ち合わせ
  - → 【客先訪問】：必ず付加される
  - → 【商品紹介】：ユーザ B〜D においてランダムに付加される
  - → 【デモ】：ユーザ B〜D においてランダムに付加される
  - → 【新規客先】：ユーザ B〜D においてランダムに付加される
- 飲食店 A，百貨店 B，コンビニ C
  - → 【バイト】：必ず付加される

同様に，持ち物は 30 個を用意した．

財布、携帯電話、名刺入れ、ノート PC、ノート PC の充電器、映像出力変換ケーブル、レーザーポインタ、筆箱、傘、着替え、歯ブラシ、クリアーファイル A、クリアーファイル B、クリアーファイル C、クリアーファイル D、クリアーファイル E、クリアーファイル F、クリアーファイル G、商品カタログ、会社紹介資料、教科書 A、教科書 B、教科書 C、教科書 D、教科書 E、教科書 F、教科書 G、教科書 H、教科書 I、教科書 J

これまで論じてきたように，提案システムでは ID だけで物を管理しており，名称を管理しない．しかし，実験をするにあたり，人間にわかりやすい名称をつけたほう

がイメージしやすいだろうとの考えからこのような名称をつけた．そのため，設定上で各事象において必要とされた物が，一般的な感覚とはずれている場合がある．あくまで ID の代わりとしての名称である．

なお，どの事象・予定タグでどの持ち物が必要であるか，という組み合わせ（正解データ）については，付録 A で一部を示す．

想定するユーザの特性は，以下の 5 種類である．

- ユーザ A
  - → 学生をイメージ
  - → 毎週同じ予定が同じ曜日に繰り返される
  - → 各事象は週に 1 回しか発生しない
- ユーザ B
  - → 学生をイメージ
  - → 平日は毎週同じ予定が同じ曜日に繰り返される
  - → 休日は基本パターンに加え，時折別の予定に変わることがある
  - → 事象によっては週に複数回発生するものがある
  - → 一部の事象に，先に規定した予定タグがつくことがある
- ユーザ C
  - → 毎週会議がある社会人をイメージ
  - → 1 週間単位で見れば同じ予定が同じ回数存在しているが，何曜日に存在するかはランダム
  - → 一部の事象に，先に規定した予定タグがつくことがある
- ユーザ D
  - → ユーザ C をベースに変化したもの
  - → 期間途中で特定の事象で必要な持ち物が変化する．変化する事象は全体の半分を目安に，ランダムに選択する
  - → 必要な持ち物推定実験 1：パラメータkの調整でのみ使用する
- ユーザ E
  - → 完全にランダムに事象が発生する

このような特性を持つユーザについて，それぞれ 130 日分の事象と持ち物データのセットを生成した．生成に際しては，個々の特性に則って，ランダムであるべき部分については乱数で，それ以外の部分については手動で生成した．生成した事象の表については，付録 B でその一部を示す．

この作成したデータのうち，最大 100 日分を演算用データとして使用し，残り 30 日分について持ち物の推測を行う．

評価については，以下のルールに則って点数を付与して比較する．

- 通知レベルが High かつ正解だった場合
  → 3 ポイント
- 通知レベルが Middle かつ正解だった場合
  → 2 ポイント
- 通知レベルが Low かつ正解だった場合
  → 1 ポイント
- 通知レベルが Low かつ不正解だった場合
  → -1 ポイント
- 通知レベルが Middle かつ不正解だった場合
  → -2 ポイント
- 通知レベルが High かつ不正解だった場合
  → -3 ポイント
- 必要な物であるにも関わらず通知が出なかった場合
  → -10 ポイント

5.2.2. 必要な持ち物推定実験 1：パラメータkの調整

まず，前日の事象にかける重み付けを決定するパラメータkを変化させた場合について実験を行った．このパラメータは，3.2.3 節で論じたように，期間中における持ち物の変化を捉えるためのものである．期間中に変化しないのであれば，kがどのような値をとっても大きな影響を受けないと予想される．

従って，期間中で事象に必要な持ち物が変化するユーザ D と，同じ事象を持ちつつも必要な持ち物が変化しないユーザ C について，パラメータkを 0.50〜1.00 の範囲で 0.01 ずつ変えながら実験を行った．

なお，この実験においては，期間中において忘れ物はなかったものとする（忘れ物率 0%）．また，データの蓄積期間は 30 日もしくは 100 日とし，期間の半分が過ぎた点（それぞれ 16 日目〜，51 日目〜）で切り替わるものとする．

5.2.2.1. 必要な持ち物推定実験 1：パラメータkの調整の結果

得られた実験結果は，図 5.4 のグラフとおりである．期間中で持ち物が変化するユーザ D に比べてユーザ C の方が常に良いポイントを得ているのは想定どおりであるが，ユーザ C，D 共に，パラメータkが 0.9 を超えると急激にポイントが低下している．

また，通知の数について分析すると，正解通知数はほとんど変化していない（

表 5.5）ものの，誤った通知の数が飛躍的に増えていることがわかった（図 5.5）.

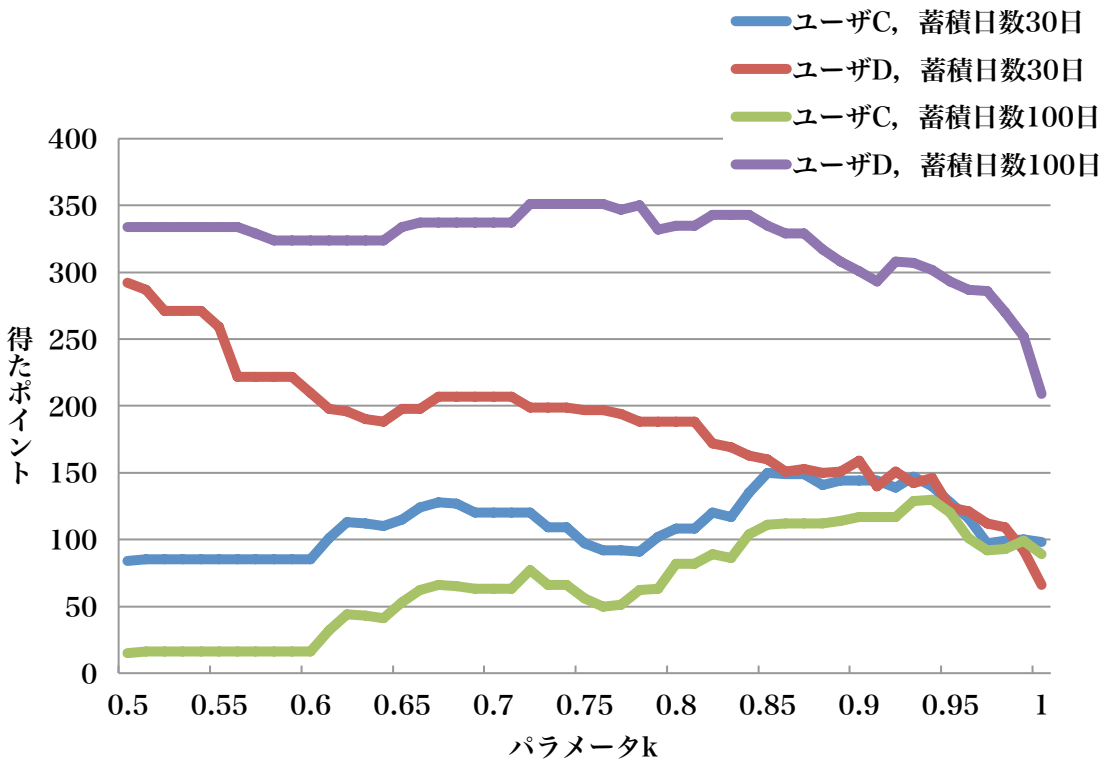


図 5.4　ユーザ C と D について，パラメータkを変化させた場合のポイント.

表 5.5　30 日間の評価期間における正しい通知の出た回数（単位：回）.

パラメータk	0.50	0.60	0.70	0.80	0.90	1.00
ユーザ C，蓄積日数 30 日	176	178	178	180	190	196
ユーザ D，蓄積日数 30 日	203	203	203	203	203	203
ユーザ C，蓄積日数 100 日	189	191	191	194	198	199
ユーザ D，蓄積日数 100 日	203	203	203	203	203	203

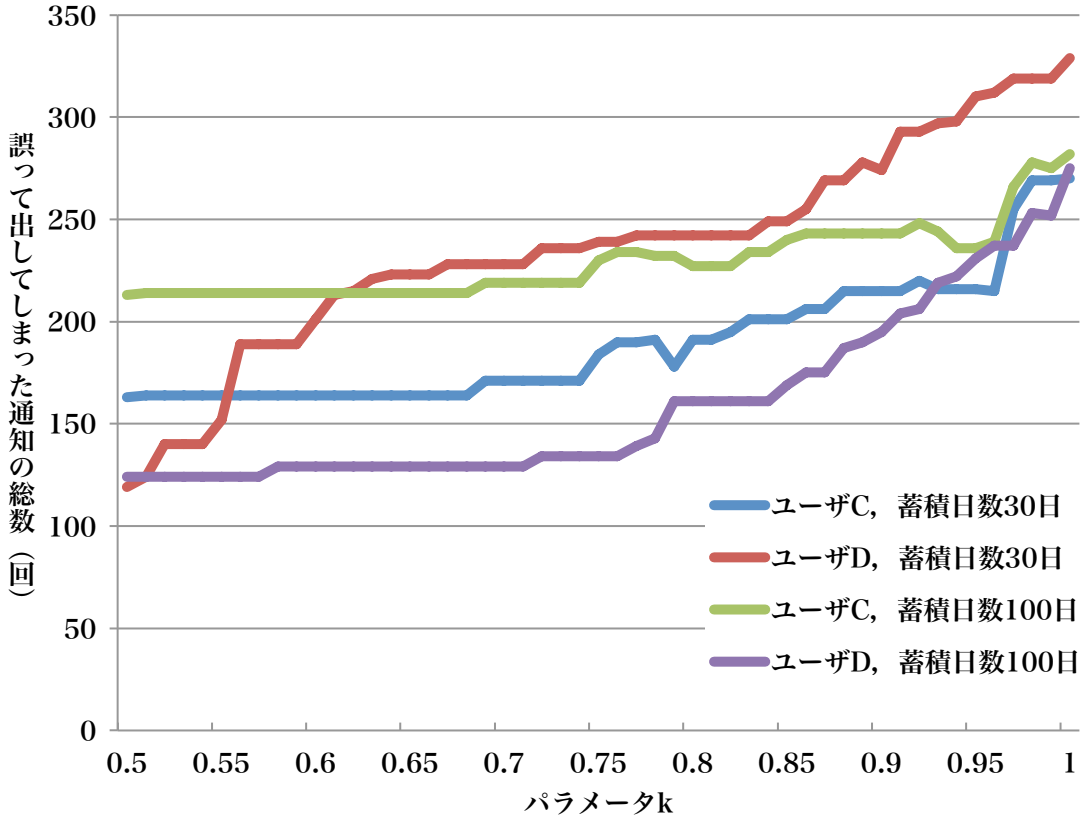


図 5.5　30 日間の評価期間における誤った通知の数の変化.

5.2.2.2.　必要な持ち物推定実験 1：パラメータkの調整の考察

図 5.4 において，蓄積日数 30 日，100 日のいずれにおいてもユーザ D の方が良いポイントを獲得するという結果となったことは非常に興味深い．期間中において持ち物が変化するということは，言わばノイズであり，ポイントも悪化するものと予測していたためである．

この，予測とのずれは，図 5.5 のグラフから一定の理由を窺い知ることができる．このグラフによれば，誤って出した通知の数がユーザ D に比べてユーザ C の方が 30～70%程度多い．すなわち，ユーザ C では誤った高いレベルの通知をしていたのに対し，ユーザ D はノイズが乗ったことで，結果的に通知のレベルが下がったと推測される．

この 2 つのグラフを合わせ見ると，$k = 0.85$ぐらいまではほとんど誤った通知が増えていないものの，ユーザ C のポイントは蓄積日数 30 日・100 日共に増加傾向にあることがわかる．これは，誤った通知のレベル（Low/Middle/High）が低下し，正し

い通知のレベルが上昇していることを意味する．

以上により，実験 3 以降ではパラメータ$k = 0.85$とすることとした．

5.2.3.　必要な持ち物推定実験 2：データ蓄積期間と推測精度

続いて，データの蓄積期間が推測に与える影響について実験した．本実験においては，物–事象関係性推測演算にて使用するデータを，30～100 日分の範囲で 10 日ずつ増やしていき，それによるポイントの変化を確認する．

この実験ではユーザ D は除かれており，ユーザ A,B,C,E について実験を行った．ユーザ D は期間中で持ち物が変化するという特性上，単純に他と期間を比較することが困難であるためである．但し，データ蓄積期間が増加した方が推測精度が良くなることは，5.2.2.1 節で載せた 2 つのグラフ（図 5.4，図 5.5）からわかる．

なお，パラメータ k は，5.1.2 節で決定した値，$k = 0.85$を使用した．

5.2.3.1.　必要な持ち物推定実験 2：データ蓄積期間と推測精度の結果

データ蓄積期間によるポイントの変化は，図 5.6 に示したグラフのとおりである．30 日から 40 日にかけては若干の変化が見られたが，それ以降は安定して変化がなかった．

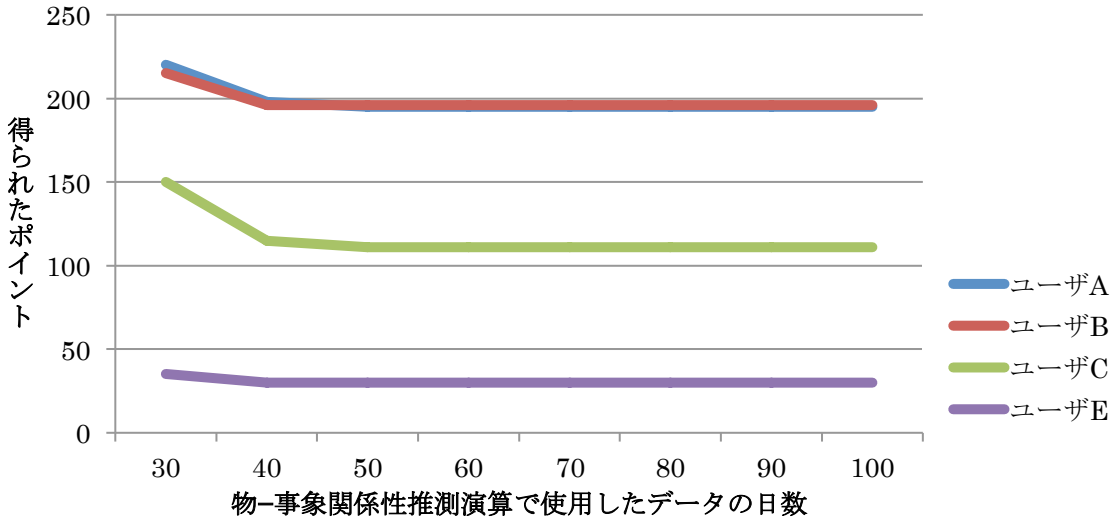


図 5.6　$k = 0.85$とした場合の推測演算に用いる日数によるポイントへの影響．

5.2.3.2.　必要な持ち物推定実験 2：データ蓄積期間と推測精度の考察

この実験からわかるように，1 ヶ月を過ぎると結果は安定し，その後は伸ばしても変化はない．そもそも，パラメータ$k = 0.85$により，30 日前のデータは大幅に減衰しており，0.85^{30}~0.076であるから，推測結果への関与は約 7.6%にまで下がっている．100 日前に至っては0.85^{100}~0であり，ほぼ存在しないに等しい．

同じ予定が 7 日おきに繰り返されるユーザ A，B に比べると，同じ予定が 7 日終期で繰り返されるものの存在する曜日がランダムなユーザ C，完全にランダムに事象が発生するユーザ E は低いポイントを示しているものの，$k = 0.85$の場合，1 ヶ月をすぎるとデータが安定することが確認できた．

5.2.4.　必要な持ち物推定実験 3：忘れ物率と推測精度

最後に，ユーザが日常的に忘れ物をしていた場合に，忘れ物の多少によって推測精度がどの程度影響を受けるかを調査した．

今回の実験においては，設定された忘れ物率（期間中において持っていくべき物に対する忘れ物の割合）を演算に用いる期間中の持ち物合計数にかけ，期間中における忘れ物の総数を決定する．そして，決定した忘れ物の総数だけ，演算に用いるデータからランダムに抜き取ることとした．そのため，忘れ物をする時期が偏るが発生する可能性があるが，全期間をとおして見れば，定められた忘れ物率に合致するようになっている．今回は，忘れ物率を 0〜10%の範囲で 1%刻みに変化させた．これは，実際の生活において，忘れ物が 10%以上の確率で起こるとは考えにくいためである（社会人に比べて圧倒的に忘れ物が多いと考えられる小学生においても，週に 3〜4 日忘れ物をする子どもは全体の 20%程度[59]だという調査結果もある．この中には意図的な忘れ物（宿題など）も含まれるとのことなので，これを除けばもっと低い割合になると考えられる）．

なお，前実験に引き続きパラメータ$k = 0.85$とし，データの蓄積期間は 30 日と 100 日の 2 種類で実験を行った．

5.2.4.1.　必要な持ち物推定実験 3：忘れ物率と推測精度の結果

実験の結果は，図 5.7，図 5.8 のグラフのようになった．ユーザ C を除き，ほとんどポイントに影響を与えていないことがわかる．ユーザ C についても，忘れ物率 0%のときが特異点になっているだけであり，それ以外を見ればほぼ影響は受けていない

といえる.

この時の正しい通知の回数をまとめたものが表 5.6，誤った通知の回数をまとめたものが図 5.9，図 5.10 である.

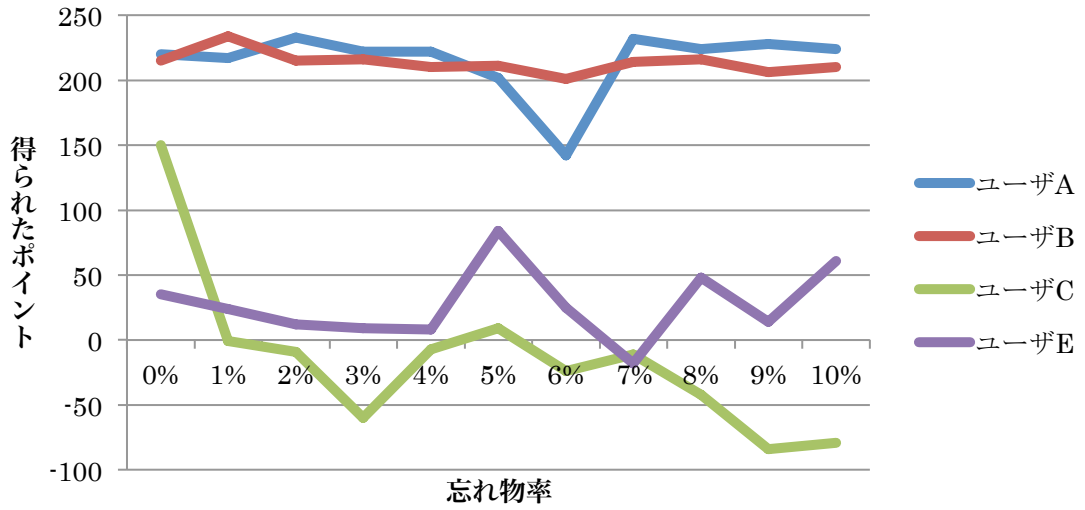


図 5.7　30 日分のデータを蓄積した場合に忘れ物率が持ち物推測に与える影響.

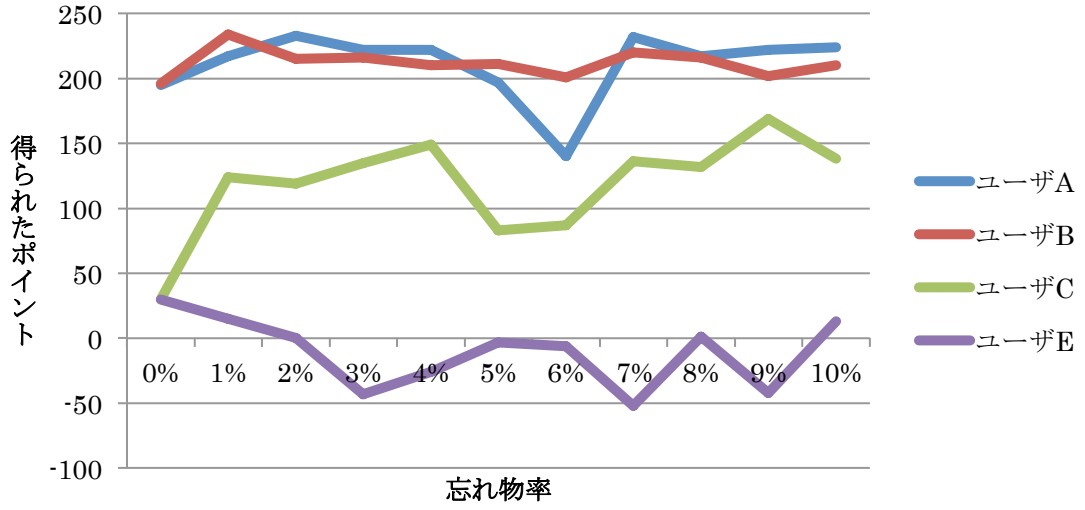


図 5.8　100 日分のデータを蓄積した場合に忘れ物率が持ち物推測に与える影響.

表 5.6 - a　忘れ物率が正しい通知の回数に及ぼした影響　0〜5%（単位：回）.

忘れ物率	0%	1%	2%	3%	4%	5%
ユーザ A，蓄積日数 30 日	109	109	109	109	109	109
ユーザ B，蓄積日数 30 日	106	106	106	106	106	106
ユーザ C，蓄積日数 30 日	191	191	189	182	189	191
ユーザ E，蓄積日数 30 日	170	169	168	167	168	169
ユーザ A，蓄積日数 100 日	109	109	109	109	109	109
ユーザ B，蓄積日数 100 日	106	106	106	106	106	106
ユーザ C，蓄積日数 100 日	198	165	165	165	165	165
ユーザ E，蓄積日数 100 日	170	169	168	167	168	169

表 5.6 - b　忘れ物率が正しい通知の回数に及ぼした影響　6〜10%（単位：回）.

忘れ物率	6%	7%	8%	9%	10%
ユーザ A，蓄積日数 30 日	100	109	109	109	109
ユーザ B，蓄積日数 30 日	106	103	106	105	106
ユーザ C，蓄積日数 30 日	185	189	185	181	187
ユーザ E，蓄積日数 30 日	170	170	170	166	169
ユーザ A，蓄積日数 100 日	100	109	109	109	109
ユーザ B，蓄積日数 100 日	106	106	106	106	106
ユーザ C，蓄積日数 100 日	160	162	160	165	165
ユーザ E，蓄積日数 100 日	170	170	170	167	170

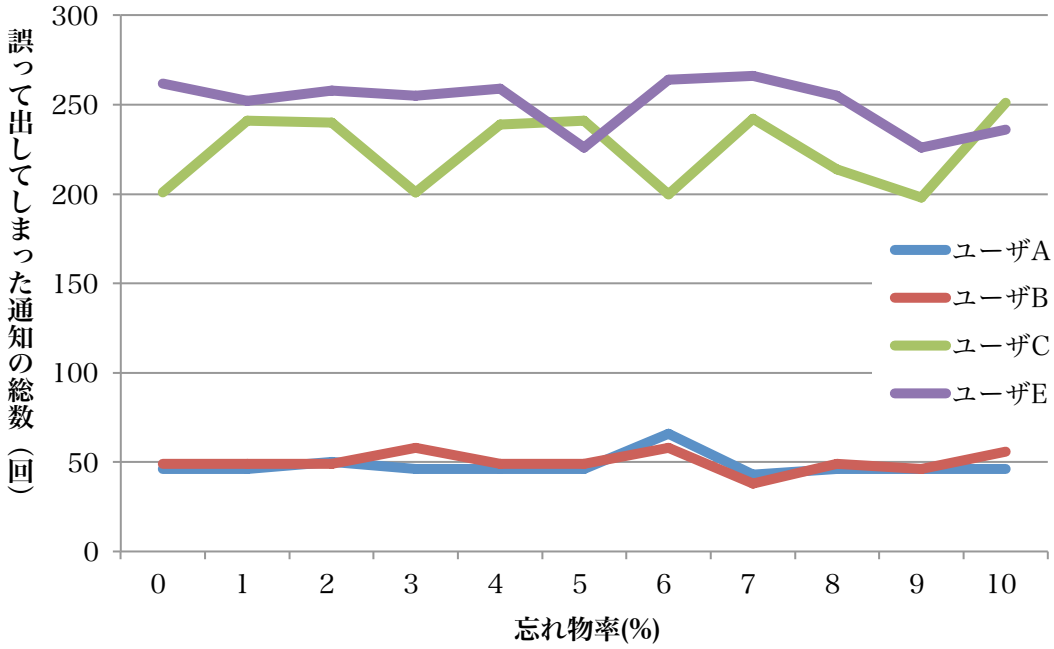


図 5.9　忘れ物率が誤った通知の回数に及ぼした影響（蓄積期間 30 日）.

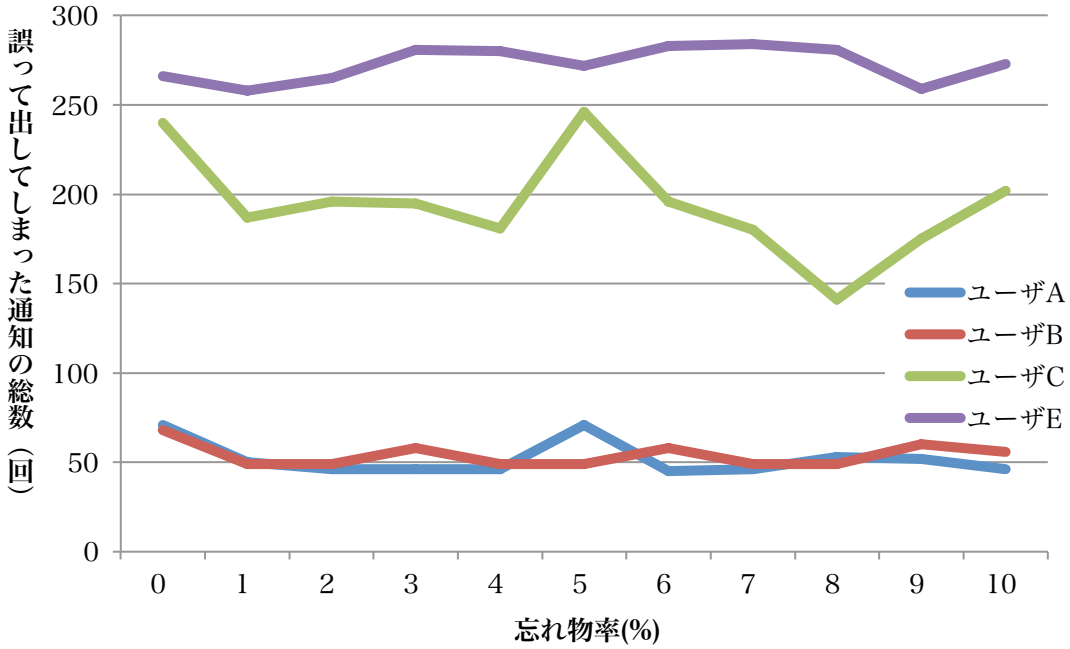


図 5.10　忘れ物率が誤った通知の回数に及ぼした影響（蓄積期間 100 日）.

5.2.4.2.　必要な持ち物推定実験 3：忘れ物率と推測精度の考察

忘れ物率の変化により，ポイントが上下に変化することが確認できた．また，正しい通知は回数にほとんど変化がなく，誤った通知の回数が上下していることから，誤った通知の方に原因があることがわかる．しかし，忘れ物率との相関性は見受けられず，図 5.7，図 5.8 から判断する限りにおいては，忘れ物率とポイントは無相関だと言える．

一方で，過去の事象ログ・持ち物ログを蓄積して推定を行っている以上，必ず物–事象関係性推測演算と必要な持ち物の推定において，忘れ物率は必ず影響を及ぼすはずである．

ここから推測されることは，通常の忘れ物の範囲（忘れ物率 10%は，10 個物を持ち出せばうち 1 つを忘れていることを意味する．実際に起こる忘れ物は，かなり低い割合だと考えられる）においては，忘れ物が及ぼす影響は軽微だということである．

5.2.5.　必要な持ち物推定実験を通した考察

ここまで，5.2.2 必要な持ち物推定実験 1：パラメータkの調整，5.2.3 必要な持ち物推定実験 2：データ蓄積期間と推測精度，5.2.4 必要な持ち物推定実験 3：忘れ物率と推測精度という 3 段階にわけて，物–事象関係性推測演算と，それを利用した持ち物推定について評価を行った．今回設定したポイントによる評価では，全体的にはプラス評価であり，一定の効果を示すことができた．

しかし，これはあくまで独自に設定したポイントによる評価であり，図 5.5 や図 5.9, 図 5.10 で示したように，ユーザへの誤通知もかなり多い．30 日間の評価で 250 回のアラートが出ているということは，1 日平均 7〜8 個の誤通知が出ているということになる．これは，3.2.4 節で設定した，相関の慣用表現を当てはめた閾値が適切でない可能性と，そもそもアルゴリズムが不充分である可能性が考えられる．

今回，主観的な要素が強すぎるために実験を行うことが出来なかったが，ユーザからシステムへのフィードバックによって，その後の関係性推測に大きく影響をおよぼし，持ち物の推定が正確に行えるようになる可能性を秘めている．

第6章 結論

6.1. まとめ

本研究では，物の名称をユーザに与えることをしなくても，「忘れ物があるか」「その忘れ物はどの事象で使う物か」という 2 点を人に提示できれば，忘れ物に対する「気付き」を誘起することができるのではないかという仮説をたて，この仮説に沿い，また仮説を確認する意味で「『気付き』を誘起する忘れ物防止支援システム」を提案した．本システムは Passive 型の RFID システムを利用するが，既存の提案と大きく異なるアプローチに基づいている．具体的には，持ち物と RFID の対応表や，いつどんな物が必要であるかというリストをユーザが作成する必要がない．過去，どのような予定や気象条件（これを本研究では「事象」と呼んでいる）があった時に，どのような物（に貼り付けられた RF タグ）が持ちだされたか，という持ち物ログとも言うべきものを蓄積することによって，持ち物と事象の関係性を推測し，未来の持ち物についても予測するアルゴリズムを提案することによって実現した．

この持ち物と事象の関係性を推測するアルゴリズムは，ベイズ統計学的な発想に基づいており，物と事象の共起確率を求める．さらに「予定タグ」と呼ぶ手法によって予定のタギングをユーザに手伝ってもらうことで新たな予定への対応力を高め，過去のデータに比べ新しいデータをより重視するためのパラメータkを導入することで期間中の持ち物変化にも追従できるようになった．

既存の手法と大きく異なる点はもう 1 つある．RFID システムを用いる点は既存手法でも散見されたものであるが，これまでの手法のように玄関にゲート型 RFID リーダを設置したりすることは必要としない．ユーザにはモバイル RFID リーダを持ち歩いて生活してもらい，システムは常時持ち物を監視する．本研究の評価実験においてはモバイル RFID リーダと，このリーダと Bluetooth 接続するスマートフォンを別々に持ち歩いたが，今後スマートフォンに RFID リーダが内蔵されれば，スマートフォンだけ持ち歩けば提案システムを使えることとなる．

以上の提案に沿ったシステムを Android スマートフォン上に実装を

評価実験においては，システムを大きく 2 つの部分にわけて性能評価を行った．まず，モバイル RFID リーダで持ち物を正確に認識することが可能であるか，実際の生活に則した形で実験を行い，必ずしも良い値ではないものの，充分に可能性を示せるだけの認識率を得た．

続いて持ち物と事象のログを数十日分システムに入力した場合に，その後の持ち物を正確に予測できるかを実験した．この実験では，パラメータkと推測演算に使うデ

ータの量，忘れ物をする確率をそれぞれ変化させ，ユーザへのアラートと正解データとのずれを評価する独自基準・ポイントが，どのように変化するかを確認した．この実験から全体的にプラス評価であることが確認でき，提案手法が一定の精度で持ち物を推測することが可能であることをを示した．

6.2. 考察・今後の展望

本研究の提案手法は，既存の提案と異なり，持ち物と RFID の対応表や，いつどんな物が必要であるかというリストをユーザが作成する必要がない．加えて玄関などへのゲート型 RFID リーダの設置という手間もない．システム導入時の手間を減らすことは，気軽に使い始めて貰えることにも繋がる．もちろん，手間だけ減らしても，そのシステムが有用でなければ意味が無い．本提案は，忘れ物への「『気付き』を誘起」しさえすれば，あとはユーザが自分で忘れ物を発見できるだろう，という仮定をすることで，手間を極力減らしながら最大限の効果を得られるように考えられている．

一方で，提案したアルゴリズムで，完全に期待通りの効果を得られているとはまだ言うことができない．実装・評価実験によって判明したように，提案手法で正しく全ての持ち物を正しく推論できているとは言えず，改良が欠かせない．持ち物リストが多くなってしまうということは，即ち必要のない物までユーザに通知してしまうということである．これでは，物の名称そのものを提示しないだけに，本来必要ないはずの持ち物再確認をユーザに強いることになってしまう．度々誤ったアラートが出ていては，ユーザにシステムを信頼してもらえず，本当に忘れ物があった場合にも，ユーザのアラート軽視によって持ち物を再確認してもらえない可能性もある．

これを軽減し，より使えるシステムとするには，2 つの手法が考えられる．まず，アルゴリズムを改良し，より高品質な関係性推測をできるようにすることである．現在は全ての日のデータを等価に扱っているが，事象の少ない日のデータはより「使える」データとして重み付けを行う仕組みを取り入れたり，その事象があった日に持ちだされていない物は確実にいらないものだ，として，「ある」日よりも「ない」日を重視するような仕組みを取り入れることが考えられる．もう 1 つの手法は，ユーザインタフェイスの改良である．現在はスマートウォッチ上に文字でアラートが出ているだけであり，通知の信頼度も背景色の変化にしか利用されていないが，専用の UI を作り，もっと差をつけてもいいだろう．例えば，より重要だと考えられるアラートでは画面上に通知が出るが，それ以外はスマートフォンの画面でリストを閲覧するようにしたり，重要なアラートではユーザが確認するまでバイブレートを続けるが，あまり信頼度が高くないと考えられるアラートの場合には一瞬震えるだけ，というような差が考えられる．まだスマートウォッチという分野は立ち上がったばかりであり，スマ

ートウォッチ上でできることは限られているが，今後はより表現力豊かな UI を使うことができるようになると考えられる．ユーザへアラートを通知する部分と，ユーザからフィードバックをする部分については，更なる検討が必要だろう．

今回，必要な物を持ち出し忘れたというタイプの忘れ物を対象に研究を行ったが，提案手法では RFID リーダをユーザが必ず持ち歩いているため，電車内などに持ち物を置いてきてしまうといった意味の忘れ物にも対応できる可能性を秘めている．また，自宅を出て会社に行って書類を持って取引先に行く，というような，より複雑な人と物の動きについてもトラッキングし，忘れ物通知を出すタイミングを調整できる可能性も秘めている．これらは，ゲート型 RFID リーダを使っていては提示することのできない価値である．そのような拡張をするためにも，基礎となるモバイル RFID リーダによる持ち物認識，そして物と事象の関係性推測を確実に行えるよう，更なる研究開発が必要である．

謝辞

まずは本研究の機会を与えてくださり，絶えず丁寧な御指導を賜りました，電気通信大学教授多田好克先生に深く感謝致します．多田先生には，普段からの御指導御鞭撻のみならず，学内外の先生方からも知恵を頂くべく問い合わせをして頂くなど，様々な形で私の研究を助けて頂きました．

同大学准教授小宮常康先生，客員准教授末田欣子先生，日本電信電話株式会社鶴岡行雄先生，客員准教授本庄利守先生にも，基盤ソフトウェア学講座の会合などの場を通し，私の至らないアイデアを膨らませて頂いたり，私だけでは思い至らないような視点から御意見を頂戴したりと，お忙しい中で様々に御指導頂きました．基盤ソフトウェア学講座の先生方のお力あっての本論文です．本当にありがとうございました．

また，このような分野で研究をするに至ったのは，電気通信大学教授市川晴久先生，助教川喜田佑介先生，インターネットマルチフィード株式会社代表取締役副社長細谷僚一先生，大阪大学特任助教神山和人先生，情報通信研究機構技術員寺田直美先生をはじめとする市川研究室関係者の皆様方あってのものであります．この場を借りて深く御礼申し上げます．

基盤ソフトウェア学講座の学生諸氏にも，日々支えていただきました．先輩である藤田竜一氏，村井栄王氏，森中翔太郎氏，片桐国建氏，石田峰文氏，神保直幸氏，若井英之氏，中原祥吾氏はもちろん，後輩のグエン クアン ヒエップ氏，木田純平氏，関湧大氏，ゾリーグ ウンダラム氏，李明元氏，新夕智啓氏，そして苦楽を共にした同期である熊谷佑弥氏，吉原大夢氏，磯谷俊明氏，工藤朋哉氏，鈴木駿介氏と，多くの学生の中で充実した２年間を過ごすことができました．深く感謝いたします．特に吉原大夢氏には，実装について沢山の助言を頂きました．重ねて御礼申し上げます．

この２年間，まだまだ多くの方にお世話になりました．八ヶ岳レジャーセンターの須田正治氏，八ヶ岳パイ工房の三井龍史氏や市川研究室 OG の阿部有希氏，同期の大渡裕太氏，熊谷啓氏．Team HLAP として共に活動した池野早紀子氏，高藤寿人氏，田口裕美氏，堀田和也氏，小野楓氏．電気通信大学人間コミュニケーション学科在籍時の友人である阿河伶太氏，内田貴之氏，小橋雄太氏，小山佳佑氏，鈴木貴暁氏，園部琢人氏，高野諒氏，丹後治也氏，中野浩道氏．安部俊孝氏，村田澪氏をはじめとした同大学生協学生委員会諸氏．岩本拓巳氏，遠山那由他氏，福井直樹氏，細淵晃平氏，山﨑純氏をはじめとした桐朋オリエンテーリング部の OB たち．ここに挙げきれないほどに多くの友人に励まされて，この論文を書きあげられました．深く感謝致します．

そしてなによりも，長きに渡って何不自由のない生活を送れるように支えてくれた家族に深く感謝し，謝辞と致します．

参考文献

[1] 農林水産省, “牛のトレーサビリティ”, http://www.maff.go.jp/j/syouan/tikusui/trace/, 農林水産省, 参照 Jan. 2015.

[2] NTT アドバンステクノロジ株式会社, “物品管理システム QpIDeal”, NTT アドバンステクノロジ株式会社, http://www.ntt-at.co.jp/product/qpideal_web/, 参照 Jan. 2015.

[3] 中央システム株式会社, “fine asset,” 中央システム株式会社, http://www.smart-works.jp/fineasset/, 参照 Jan. 2015.

[4] 株式会社インフュージョン, “実地棚卸システム 在庫スイート 3 棚卸”, 株式会社インフュージョン, http://www.infusion.co.jp/suite3/tana/, 参照 Jan. 2015.

[5] 株式会社デンソーウェーブ, “QR ぱんだ de 棚卸”, 株式会社デンソーウェーブ, http://www.denso-wave.com/ja/adcd/app/panda-st/, 参照 Jan. 2015.

[6] 三菱電機コントロールソフトウェア株式会社, “RFID 対応 e!Tracking”, 三菱電機コントロールソフトウェア株式会社, http://www.mcr.co.jp/products/management/etracking.html, 参照 Jan. 2015.

[7] 株式会社東北システムズ・サポート, “資産・物品管理システム MONISTOR”, 株式会社東北システムズ・サポート, http://www.tss21.co.jp/product/package/monistor/, 参照 Jan. 2015

[8] 高千穂交易株式会社, “RFID 棚卸・資産管理”, http://www.takachiho-kk.co.jp/prod/retail_secu/rfid/, 高千穂交易株式会社, 参照 Jan. 2015.

[9] ソニー株式会社, “非接触 IC カード技術 Felica”, ソニー株式会社, http://www.sony.co.jp/Products/felica/, 参照 Jan. 2015.

[10] 庄木裕樹, “ワイヤレス電力伝送技術の実用化に向けた課題と取り組み”, 株式会社東芝 研究開発センター, http://cic-infonet.jp/section/activity/pdf/121207_2.pdf, Dec. 2012.

[11] 塚本昌彦, 板生 知子, “ウェアラブルコンピューティングとユビキタスサービス”, オペレーションズ・リサーチ：経営の科学, vol. 49, no. 4, pp. 210-216, Apr. 2004.

[12] Oculus VR Inc., “Oculus Rift - Virtual Reality Headset for 3D Gaming,” Oculus VR Inc., https://www.oculus.com, 参照 Jan. 2015.

[13] Google Inc., “Google Glass,” Google, https://www.google.com/glass/start/, 参照 Jan. 2015.

[14] セイコーエプソン株式会社, “MOVERIO”, セイコーエプソン株式会社, http://www.epson.jp/products/moverio/, 参照 Jan. 2015.
[15] ソニー株式会社, “スマートウェア 製品一覧”, ソニー株式会社, http://www.sonymobile.co.jp/product/smartwear/, 参照 Jan. 2015.
[16] サムスン電子株式会社, “SAMSUNG Gear”, サムスン電子株式会社, http://www.samsung.com/jp/consumer/mobilephone/gear/, 参照 Jan. 2015.
[17] JAWBONE, “Jawbone ウェアラブル”, Aliph Inc., https://jawbone.com/wearable, 参照 Jan. 2015.
[18] Apple Inc., “Apple Watch,” Apple Inc., https://www.apple.com/watch/overview/, 参照 Jan. 2015.
[19] 富士通フロンテック株式会社, “製品カタログ 書類管理用ラベルタグ”, 富士通フロンテック株式会社, http://www.fujitsu.com/downloads/RETAIL/frontech/services/products/rfid/downloads/document.pdf, 参照 Jan. 2015.
[20] 日本経済新聞, “日立製作所と KDDI が携帯電話に内蔵できる RFID リーダーを開発”, 日本経済新聞, http://www.nikkei.com/article/DGXNASFK1201O_S0A710C1000000/, Jul. 2010.
[21] PHYCHIPS Ltd., “ARETE POP,” PHYCHIPS Ltd., http://arete-mobile.com/arete_pop.html, 参照 Jan. 2015.
[22] 富士通フロンテック株式会社, “UHF 帯新周波数に対応した書類管理用ラベルタグの価格半減を実現!!”, 富士通フロンテック株式会社, http://www.fujitsu.com/jp/group/frontech/resources/news/press-releases/2012/0910.html, Sep. 2012.
[23] N. Priyantha, A. Miu, H. Balakrishnan and S. Teller, “The Cricket Compass for Context-aware Mobile Applications,” Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 1-14, Jul. 2001.
[24] R. Want, A. Hopper, V. Falcao and J. Gibbons, “The Active Badge Location System,” ACM Transactions on Information Systems, vol. 10, issue 1, pp. 91-102, Jan. 1992.
[25] 服部聖彦, 中村恭平, 中嶋信生, 藤井哲也, 門洋一, 張兵, 高玉圭樹, “ユビキタスネットワークにおける位置依存サービスのための高精度位置推定システムの提案と検証”, 信学論(B), vol. J94-B, no. 10, pp. 1341-1350, Nov. 2011.
[26] 山本峻丸, “ペアリングタグによるモノ管理支援システム”, 平成 24 年度電気通信

大学電気通信学部人間コミュニケーション学科卒業論文, Feb. 2013.
[27] 山本友紀子, 石井健太郎, 今井倫太, 中臺一博, “探し物支援のための超音波を用いた誘導システム CoCo”, ヒューマンインターフェースシンポジウム, pp. 1049-1054, Spt. 2007.
[28] 中田豊久, 金井秀明, 國藤進, “スポットライトを用いた屋内での捜し物発見支援システム”, 情処学論, vol. 48 no. 12, pp. 3962-3976, Dec. 2007.
[29] 西原秀明, 窪田裕介, 村川友章, 芳賀博英, 金田重郎, “焦電センサと RFID による室内向け物品位置検出手法”, 情処学全国大会, 6V-1, Mar. 2007.
[30] 佐竹聡, 今井倫太, 川島英之, 安西祐一郎, “Brownie：カメラ上に指定された過去のランドマーク情報に基づく実世界探し物検索システム”, 日本知能情報ファジィ学会誌 知能と情報, vol. 19 no. 5, pp. 556-569, Oct. 2007.
[31] 小松崎瑞穂, 塚田浩二, 椎尾一郎, “BoxFinder：2 次元コードと写真を利用したモノ探し支援システム”, 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, no. 67, pp. 213-215, Dec. 2009.
[32] 中川真紀, 塚田浩二, 椎尾一郎, “箱ブラウザ：収納箱の手軽な撮影と閲覧システム”, ヒューマンインターフェースシンポジウム 2008 論文集, pp. 1039-1042, Spt. 2008.
[33] 上岡隆宏, 河村竜幸, 河野恭之, 木戸出正継, “I’m Here!:物探しを効率化するウェアラブルシステム”, ヒューマンインターフェース学会論文誌, vol. 6 no. 3, pp. 19-30, Aug. 2004.
[34] 日本電気株式会社, “物品管理システム SmartAsset”, 日本電気株式会社, http://www.nec.co.jp/solution/engsl/pro/smartasset/index.html, 参照 Jan. 2015.
[35] 日本電気株式会社, “UHF 帯 RFID タグ約 80 万枚を用いた精緻な販売・在庫管理を実現”, 日本電気株式会社, http://www.nec.co.jp/press/ja/1111/0803.html, Nov. 2011.
[36] 株式会社日立製作所, “日立無線 LAN 位置検知システム AirLocation II”, 株式会社日立製作所, http://www.hitachi.co.jp/products/it/network/catalog/pdf/wirelessinfo/al.pdf, 参照 Jan. 2015.
[37] Mik Lamming and Mike Flynn, “"Forget-me-not" Intimate Computing in Support of Human Memory,” Intimate Computing in Support of Human Memory Invited keynote paper for FRIEND21 Symposium on Next Generation Human Interface, pp. 125-128, Feb. 1994.

[38] Bradley J. Rhodes and Thad Starner, “The Remembrance Agent: A continuously running automated information retrieval system,” The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology, pp. 487-495, Apr. 1996.

[39] Roy Want, Andy Hopper, Veronica Falcão and Jonathan Gibbons, “The active badge location system,” ACM Transactions on Information Systems, vol.10, issue 1, pp. 91-102, Jan. 1992.

[40] Gaetano Borriello, Waylon Brunette, Matthew Hall, Carl Hartung and Cameron Tangney, “Reminding About Tagged Objects Using Passive RFID,” Ubiquitous Computing 2004, pp. 36-53, Sep. 2004.

[41] Hui-Huang Hsua, Cheng-Ning Leea, Jason C. Hungb and Timothy K. Shihc, “Smart object reminders with RFID and mobile technologies,” Mobile Information Systems, vol. 7, no. 4, pp. 317-327, Nov. 2011.

[42] 照屋のぞみ, 鈴木大作, 正木忠勝, “小学校向け学校生活支援システムの開発 -NFC を用いた登校準備機能-”, 情報システム教育学会 2013 年度学生研究発表会, Mar. 2014.

[43] 浜野悠介, “重量センサを用いた忘れ物防止のための持ち物の組み合わせ推定手法”, 平成 24 年度筑波大学情報学群情報科学類卒業研究論文, Feb. 2013.

[44] Tiancheng Zhang, Xiaonan Yu, Dejun Yue, Yu Gu and Ge Yu, “Study of an RFID Based Object-Relationship Recognition System,” The 8th International Conference on Computational Intelligence and Security, pp. 359-363, Nov. 2012.

[45] Laurie Sullivan, “RFID Implementation Challenges Persist, All This Time Later,” Information Week, Oct 2005.

[46] Barnaby J. Feder, “Despite Wal-Mart’s Edict, Radio Tags Will Take Time,” New York Times, Dec. 2004.

[47] 萩原大輔, 井上創造, 安浦寛人, “RFID 情報システムにおけるシステムレベルでの信頼性向上”, 情処学論, vol. 46, no. SIG 8(TOD 26), pp. 37-47, Jun. 2005.

[48] Jeffery, S.R., Alonso G., Franklin M.J., Wei Hong Hong and Widom J., “A Pipelined Framework for Online Cleaning of Sensor Data Streams,” ICDE ’06, pp. 140-142, Mar. 2006.

[49] Gupta A. and Sarivastava M., “Developing Auto-ID Solutions Using Sun Java System RFID Software,” Oracle Co., Oct. 2004.

[50] Shawn R. Jeffery, Minos Garofalakis and Michael J. Franklin, “Adaptive Cleaning for RFID Data Streams,” VLDB ’06, pp. 163-174, Mar. 2006
[51] Google Inc., “Google Calendar API,” Google Inc., https://developers.google.com/google-apps/calendar/, 参照 Jan. 2015.
[52] Livedoor, “Weather Hacks”, LINE 株式会社, http://weather.livedoor.com/weather_hacks/, 参照 Jan. 2015.
[53] 株式会社東北システムズサポート, “UHF 帯 RFID リーダライタ DOTR-900J シリーズ”, 株式会社東北システムズサポート, 参照 Jan. 2015.
[54] Android Developers, “Activity | Android Developers,” Google Inc., http://developer.android.com/reference/android/app/Activity.html, 参照 Jan. 2015.
[55] 理音伊織, “Android で TimePicker プリファレンスを使う,” 理音伊織, http://relog.xii.jp/mt5r/2010/09/androidtimepicker.html, 参照 Jan. 2015.
[56] Michael Pardo, “Activeandroid by Pardo,” Michael Pardo, http://www.activeandroid.com, 参照 Jan. 2015.
[57] Omni-ID Ltd., “IQ Range - Discrete Labels,” Omni-ID Ltd., http://www.omni-id.com/pdfs/Omni-ID_IQ_Range_Discrete_Labels.pdf, 参照 Jan. 2015.
[58] 総務省統計局, “日本統計年鑑　第 1 章 国土・気象 1-10　気象官署別日照時間, 天気日数”, 総務省統計局, http://www.stat.go.jp/data/nenkan/01.htm, 参照 Jan. 2015.
[59] ベネッセ教育総合研究所, “初等中等教育研究室　調査・研究データ モノグラフ・小学生ナウ”, ベネッセホールディングス, http://berd.benesse.jp/shotouchutou/research/detail1.php?id=3418 , 1989.

付録 A.　実験で用いた物と事象の対照表の一部

	財布	携帯電話	名刺入れ	ノート PC	筆箱	傘
Daily	1	1				
【発表】				1	1	
【会議】				1	1	
【講義】				1	1	
【試験】					1	
【客先訪問】					1	
【商品紹介】			1			
【新規客先】			1			
【デモ】			1			
【バイト】			1			
雨						
晴						1
講義 A						
講義 B					1	
講義 C					1	
講義 D				1	1	
ゼミ				1	1	
部内会議			1	1	1	
社内会議			1			
A 社打ち合わせ			1		1	
B 社打ち合わせ			1		1	
C 社打ち合わせ			1		1	
D 社打ち合わせ			1		1	
飲食店 A						
百貨店 B					1	
コンビニ C					1	
飲み会			1			
ライブ						

付録 B.　実験で用いたユーザ毎の予定表

以下に，実験で用いたユーザ毎の予定表の一部を示す．実験においては，事象収集機構をバイパスするため，このような表を CSV ファイルとして入力して一気に計算できるメソッドを作成した．

- **ユーザ A**

1 日目	Daily	雨	【バイト】	飲食店 A	百貨店 B		
2 日目	Daily	雨	【講義】	講義 A	講義 F	講義 H	
3 日目	Daily	晴	【講義】	講義 B			
4 日目	Daily	晴	【講義】	講義 C	講義 G		
5 日目	Daily	雨	【講義】	講義 D	【バイト】	コンビニ C	
6 日目	Daily	晴	【講義】	講義 E			
7 日目	Daily	雨	【バイト】	飲食店 A			
8 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
9 日目	Daily	晴	【講義】	講義 A	講義 F	講義 H	
10 日目	Daily	雨	【講義】	講義 B			
11 日目	Daily	雨	【講義】	講義 C	講義 G		
12 日目	Daily	雨	【講義】	講義 D	【バイト】	コンビニ C	
13 日目	Daily	雨	【講義】	講義 E			
14 日目	Daily	晴	【バイト】	飲食店 A			
15 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
16 日目	Daily	雨	【講義】	講義 A	講義 F	講義 H	
17 日目	Daily	晴	【講義】	講義 B			
18 日目	Daily	晴	【講義】	講義 C	講義 G		
19 日目	Daily	晴	【講義】	講義 D	【バイト】	コンビニ C	
20 日目	Daily	晴	【講義】	講義 E			
21 日目	Daily	晴	【バイト】	飲食店 A			
22 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
23 日目	Daily	晴	【講義】	講義 A	講義 F	講義 H	
24 日目	Daily	雨	【講義】	講義 B			
25 日目	Daily	晴	【講義】	講義 C	講義 G		
26 日目	Daily	雨	【講義】	講義 D	【バイト】	コンビニ C	
27 日目	Daily	雨	【講義】	講義 E			

● **ユーザ B**

1 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
2 日目	Daily	晴	【講義】	講義 A	講義 F	講義 H	
3 日目	Daily	晴	【講義】	講義 B			
4 日目	Daily	雨	【講義】	講義 C	講義 G		
5 日目	Daily	晴	【講義】	講義 D	【バイト】	コンビニ C	
6 日目	Daily	雨	【講義】	講義 E			
7 日目	Daily	晴	【バイト】	飲食店 A			
8 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
9 日目	Daily	晴	【講義】	講義 A	講義 F	講義 H	
10 日目	Daily	晴	【講義】	講義 B			
11 日目	Daily	雨	【講義】	講義 C	講義 G	【試験】	
12 日目	Daily	雨	【講義】	講義 D	【バイト】	コンビニ C	
13 日目	Daily	晴	【講義】	講義 E			
14 日目	Daily	晴	【バイト】	飲食店 A			
15 日目	Daily	晴	【バイト】	飲食店 A	飲み会		
16 日目	Daily	晴	【講義】	講義 A	講義 F	講義 H	
17 日目	Daily	晴	【講義】	講義 B			
18 日目	Daily	雨	【講義】	講義 C	講義 G		
19 日目	Daily	雨	【講義】	講義 D	【バイト】	コンビニ C	
20 日目	Daily	晴	【講義】	講義 E			
21 日目	Daily	晴	【バイト】	飲食店 A			
22 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
23 日目	Daily	雨	【講義】	講義 A	講義 F	講義 H	
24 日目	Daily	晴	【講義】	講義 B			
25 日目	Daily	雨	【講義】	講義 C	講義 G		
26 日目	Daily	晴	【講義】	講義 D	【バイト】	コンビニ C	
27 日目	Daily	雨	【講義】	講義 E			
28 日目	Daily	晴	【バイト】	飲食店 A			
29 日目	Daily	晴	【バイト】	飲食店 A	百貨店 B		
30 日目	Daily	雨	【講義】	講義 A	講義 F	講義 H	【試験】
31 日目	Daily	晴	【講義】	講義 B			
32 日目	Daily	晴	【講義】	講義 C	講義 G		
33 日目	Daily	晴	【講義】	講義 D	【バイト】	コンビニ C	

● ユーザ C

1 日目	Daily	雨	買い物	説明会			
2 日目	Daily	晴					
3 日目	Daily	晴	C 社打ち合わせ	【客先訪問】	【デモ】		
4 日目	Daily	晴	B 社打ち合わせ	【客先訪問】	【商品紹介】	出張	
5 日目	Daily	晴	部内会議	【会議】	【発表】	A 社打ち合わせ	【客先訪問】
			【デモ】	【商品紹介】			
6 日目	Daily	晴					
7 日目	Daily	晴	D 社打ち合わせ	【客先訪問】	【商品紹介】	【デモ】	
8 日目	Daily	晴	社内会議	【発表】	【会議】	部内会議	【会議】
9 日目	Daily	晴	買い物				
10 日目	Daily	雨	C 社打ち合わせ	【客先訪問】	A 社打ち合わせ	【客先訪問】	【商品紹介】
11 日目	Daily	晴	出張	説明会			
12 日目	Daily	晴					
13 日目	Daily	晴					
14 日目	Daily	晴	社内会議	【発表】	【会議】	C 社打ち合わせ	【客先訪問】
			【デモ】	【商品紹介】			
15 日目	Daily	雨	D 社打ち合わせ	【客先訪問】			
16 日目	Daily	雨	出張				
17 日目	Daily	晴					
18 日目	Daily	晴	買い物	説明会			
19 日目	Daily	雨	B 社打ち合わせ	【客先訪問】	A 社打ち合わせ	【客先訪問】	【商品紹介】
20 日目	Daily	晴					
21 日目	Daily	晴	D 社打ち合わせ	【客先訪問】	社内会議	【発表】	【会議】

● ユーザ D

1 日目	Daily	晴					
2 日目	Daily	晴	B 社打ち合わせ	【客先訪問】	買い物		
3 日目	Daily	晴	社内会議	【発表】	【会議】	C 社打ち合わせ	【客先訪問】
			説明会	【商品紹介】			
4 日目	Daily	雨					
5 日目	Daily	雨	C 社打ち合わせ	【客先訪問】	説明会	【商品紹介】	
6 日目	Daily	晴	出張				
7 日目	Daily	晴	買い物	部内会議	【会議】	【発表】	A 社打ち合わせ
			【客先訪問】				
8 日目	Daily	晴					
9 日目	Daily	晴					
10 日目	Daily	晴					
11 日目	Daily	晴					
12 日目	Daily	晴	買い物	出張	部内会議	【会議】	
13 日目	Daily	晴	C 社打ち合わせ	【客先訪問】	B 社打ち合わせ	【客先訪問】	【商品紹介】
14 日目	Daily	晴	D 社打ち合わせ	【客先訪問】			
15 日目	Daily	晴					
16 日目	Daily	雨	社内会議	【発表】	【会議】	A 社打ち合わせ	【客先訪問】
			説明会				
17 日目	Daily	晴					
18 日目	Daily	晴					
19 日目	Daily	雨	部内会議	【会議】			
20 日目	Daily	晴	D 社打ち合わせ	【客先訪問】			
21 日目	Daily	雨	社内会議	【発表】	【会議】		

● **ユーザ E**

1 日目	Daily	晴	D 社打ち合わせ	飲み会	百貨店 B	講義 B	講義 A
			【客先訪問】	【新規客先】	【講義】		
2 日目	Daily	晴	講義 H	説明会	D 社打ち合わせ	コンビニ C	【バイト】
			【講義】	【客先訪問】	【商品紹介】		
3 日目	Daily	晴	百貨店 B	B 社打ち合わせ	【バイト】	【客先訪問】	【新規客先】
			【商品紹介】				
4 日目	Daily	晴	ライブ	講義 G	百貨店 B	説明会	【講義】
			【バイト】				
5 日目	Daily	晴	飲食店 A	飲み会	コンビニ C	【バイト】	
6 日目	Daily	晴	B 社打ち合わせ	講義 G	B 社打ち合わせ	講義 C	【講義】
			【客先訪問】	【デモ】			
7 日目	Daily	雨	飲み会	講義 H	飲食店 A	C 社打ち合わせ	【講義】
8 日目	Daily	晴	講義 B	【講義】	【試験】	【発表】	
9 日目	Daily	晴	説明会				
10 日目	Daily	雨	コンビニ C	買い物	【バイト】		
11 日目	Daily	晴					
12 日目	Daily	晴	D 社打ち合わせ	【客先訪問】	【デモ】		
13 日目	Daily	晴	講義 E	【講義】	【発表】		
14 日目	Daily	晴	D 社打ち合わせ	ライブ			
15 日目	Daily	雨	ライブ	講義 F	【講義】		
16 日目	Daily	晴	講義 B	講義 F	講義 B	社内会議	【講義】