Vol.2014-UBI-42 No.8
2014/5/29

Application as a Sensor (AaaS) 手法によるユーザ注意の検知

大越 匡[1,a)]　中澤 仁[2]　徳田 英幸[2]

概要：本研究は，ユーザが何らかのアプリケーション操作をともなってユビキタスコンピューティングにおける目標行動を行う環境における，“Application as a Sensor” (AaaS) アプローチに基づくユーザの注意状態の検知を提案する．本システムは，ユーザの注意に対するシステムの適応的動作実現のため，ユーザの現在の注意対象と “breakpoint”? を，ユーザが操作する複数のデバイス上のアプリケーションそれぞれに特化した情報を使って検知する．またその検知において，モバイルデバイス上でリアルタイムに，生体センサを用いずに実現する点を特徴とする．

キーワード：注意 (attention)，モバイルセンシング，ユビキタスコンピューティング

Application as a Sensor (AaaS) Approach for User Attention Sensing

1. Introduction

ユビキタスコンピューティング時代の到来とともにユーザへもたらされる情報の量が増え続ける一方，ユーザの「注意」の量が基本的には変わらない事から，マルチタスク環境における “通知” によるユーザへの “割り込み” は，次第に大きな問題として認知されつつある．

ユーザが所有するコンピューティングデバイスの数は，家やオフィス，街中などユーザを取り巻く環境内ににおけるそれと同様，増加の一途を辿っている．多くのユーザは最近，スマートフォン，タブレット，ノート PC，ウェアラブルデバイス，デジタルカメラなど，より多くのネットワーク接続されたモバイルデバイス?を所有する．ユーザはまた，しばしば，それらの所有デバイスに加え，公共ディスプレイやキオスク端末など公共空間に設置されたデバイスも含めて，複数のデバイスを同時に操作する?場合がある．また同時に，Web のプログラマビリティ向上，成熟するミドルウェアやクラウド技術を背景とした迅速なサービス開発，所謂 “AppStore” を通じたグローバルなアプリケーション配信などの影響もあり，各ユーザが利用するアプリケーション，サービス，コミュニケーションチャネルの数も増えている．更には，email や SMS などの従来型のコミュニケーション手段に加えて，ソーシャルネットワーク Web サービスが搭乗したことにより，ユーザはより多くの他のユーザと接続しやりとりを行う傾向にある．

このような背景において，限りある “人間の注意” のリソースは、コンピューティングにおける新しいボトルネックであると指摘?されている．人間ユーザの視点から見れば，過度に多い情報がもたらされる状況は，広義に「情報過多」として知られる．

本研究では特に，割り込み過多 *(interruption overload)* 現象，言い換えれば，コンピュータシステムからの過剰量の通知やその不適切な配送による割り込みによって引き起こされる「ユーザの注意の散漫」に着目する．そのうえで本研究では，ユーザの割り込み過多を解消をめざし，ユーザの注意状態に適応的なシステム側での通知動作を実現するため，ユーザの注意の検出する仕組みを提案する．

ユーザへの様々な通知を即座に配送する典型的な通知システムは，ユーザの生産性に負の影響を与えることが先行研究?における管理下試験で明らかになっている．本研究は，ユーザが何らかのアプリケーション操作をともなってユビキタスコンピューティングにおける目標行動を行う環

[1] 慶應義塾大学大学院 政策・メディア研究科
Graduate School of Media and Governance, Keio University
[2] 慶應義塾大学 環境情報学部
Faculty of Environment and Information Studies, Keio University
[a)] slash@ht.sfc.keio.ac.jp

境における，“Application as a Sensor” (AaaS) アプローチに基づくユーザの注意状態の検知を提案する．本システムは，ユーザの生産性を保つシステムの適応的動作実現のため，ユーザの現在の注意対象と “breakpoint”? を，ユーザが操作する複数のデバイス上のアプリケーションそれぞれに特化した情報を使って検知する．またその検知において，モバイルデバイス上でリアルタイムに，生体センサを用いずに実現する点を特徴とする．これは関連する他の研究が，主に研究所内のデスクトップ環境を焦点とし，生体センサを使用して，遡及的な検知を行うのとは対照的である．

本研究の貢献は，マルチデバイス環境において，専用の生体センサを用いることなく，実時間にユーザの注意状態を検知するシステムの設計を提案する点にある．

2. 割り込み過多

マルチタスク環境における過剰量の通知によるユーザへの “割り込み過多” は，限りある “人間の注意” のリソースがコンピューティングにおける新しいボトルネックであると考えられることから，情報過多に関する研究の中でも次第に注目を集めている．近年，Interruptions and Multitasking (IM) という分野において多くの研究?. が行われている．

割り込み過多の主な要因はコンピューティングシステムからの通知である．通知はもともと，情報をユーザの注意の焦点の外側から，より迅速にまた適時的にユーザへ提供するために設計された仕組みである．しかしそういった便益にも関わらず，ユーザへの様々な通知を即座に配送する典型的な通知システムは，ユーザの生産性に負の影響を与えることが先行研究?によって明らかになっている．また他の研究?は，ユーザがその時遂行している主たるタスクと通知情報の間にある文脈の違いが，もうひとつの原因であることを明らかにしている．

通知は，実際のシステムにおいてはユーザによって設定可能であり無効化することも可能であるが，単純に無効化する事は通知自身の便益を打ち消し，ユーザの「適時的な情報の受信」のニーズを満たせなくなることを意味する．実際に，Iqbal?らの研究では，多くのユーザが，単純に通知システムを無効化して手動で情報をチェックするよりも，割り込みのコストを考慮に入れても通知による情報配信を好んだ．

通知がユーザによって知覚・認識されるとき，ユーザの注意リソースのうちいくらかが，通知で配送された情報へと割り当てられる．これが「割り込み」が発生する瞬間である．人間の注意は有限のリソース?であるので，注意のいくらかを通知へ分割して割り当てることは，“devided attention (DA)” という状態を作り出す．そしてこの DA 状態は，人間の記憶能力に大きな負の影響を与える?ことが知られている．従って通知が引き金となって生じる DA 状態においては，ユーザが現在行っている主要なタスク (primary task) の効率が容易に影響を受ける．

こういったユーザの割り込み過多状態を緩和するため，現在のユーザの注意状態や通知される情報に応じて動的に通知のタイミングやメディア，内容を調節する「適応的な通知サポート」が明らかに必要とされる．本研究は，適応的な通知サポートの中でも，“いつ通知を行えば良いか”，“どういった情報を通知すれば良いか” の推定のために，ユーザの注意状態の検知に焦点をあてる．

1 章で紹介した近年のユビキタスコンピューティングの流れにおける「通知」の特徴として，下記を本研究の焦点とする．

- **通知元や通知の種類の多様化:** 例えば，近年急速に広がりつつあるソーシャルネットワーク上の他ユーザからの情報更新や，参加型センシング?におけるセンシングの依頼などがあげられる．
- **緊急度合いの多様化:** 受信したユーザは数秒以内に物理的な行動を起こす必要がある緊急地震速報?は，新しい通知の「緊急度合い」を実装する例としてあげられる．
- **終日継続する割り込み過多:** ユビキタス環境においてユーザは，常にモバイルデバイスを持ち歩き，ユビキタスサービスを日常的 (朝起きてから夜寝るまで．時には寝ている間も) に利用する．従って「割り込み過多」はユーザの生活に密着して終日継続する．

また本研究では，

ユーザの ” ユビキタスコンピューティングにおける，何らかの目的をもった対話的なタスク” の間における割り込みを対象とする．ドキュメント編集，ビデオの閲覧，e-book の閲覧，ゲームのプレイなどが例としてあげられるこの種のタスクは，情報空間の何らかのコンテンツの操作 (読み込み or 書き込み) を含む．これは筆者らが，ユビキタスコンピューティングにおいて，不適切な割り込みの脅威にさらされるような注意が必要とされるタスクの相当部分は，上に述べたような何らかの情報空間内のコンテンツ操作に関係すると考えるからである．この対象には，オフラインでの (物理的な本を読む) 読書や料理といったタスクは含まれない．しかしながら筆者らは，本研究における問題領域の単純化の第一段階としては，このような対象の絞り込みは適切だと考える．

3. アプリケーション層の情報を使った注意の検知

“いつ通知を行えば良いか”，“どういった情報を通知すれば良いか” の推定に向けて，本章では，アプリケーション層からの情報を利用したユーザ注意状態の検知について述べる．具体的には，ユーザが操作する複数のモバイル機

正誤表

提出した原稿（PDF）に不備がありましたので、次ページ以降のとおり訂正いたします。

Application as a Sensor (AaaS) 手法によるユーザ注意の検知

大越 匡[1,a)]　中澤 仁[2]　徳田 英幸[2]

概要：本研究は，ユーザが何らかのアプリケーション操作をともなってユビキタスコンピューティングにおける目標行動を行う環境における，“Application as a Sensor” (AaaS) アプローチに基づくユーザの注意状態の検知を提案する．本システムは，ユーザの注意に対するシステムの適応的動作実現のため，ユーザの現在の注意対象と “breakpoint”[19] を，ユーザが操作する複数のデバイス上のアプリケーションそれぞれに特化した情報を使って検知する．またその検知において，モバイルデバイス上でリアルタイムに，生体センサを用いずに実現する点を特徴とする．

キーワード：注意 (attention)，モバイルセンシング，ユビキタスコンピューティング

1. はじめに

ユビキタスコンピューティング時代の到来とともにユーザへもたらされる情報の量が増え続ける一方，ユーザの「注意」の量が基本的には変わらない事から，マルチタスク環境における “通知” によるユーザへの “割り込み” は，次第に大きな問題として認知されつつある．

ユーザが所有するコンピューティングデバイスの数は，家やオフィス，街中などユーザを取り巻く環境内におけるそれと同様，増加の一途を辿っている．多くのユーザは最近，スマートフォン，タブレット，ノート PC，ウェアラブルデバイス，デジタルカメラなど，より多くのネットワーク接続されたモバイルデバイス [6] を所有する．ユーザはまた，しばしば，それらの所有デバイスに加え，公共ディスプレイやキオスク端末など公共空間に設置されたデバイスも含めて，複数のデバイスを同時に操作する [9] 場合がある．また同時に，Web のプログラマビリティ向上，成熟するミドルウェアやクラウド技術を背景とした迅速なサービス開発，所謂 “AppStore” を通じたグローバルなアプリケーション配信などの影響もあり，各ユーザが利用するアプリケーション，サービス，コミュニケーションチャネルの数も増えている．更には，email や SMS などの従来型のコミュニケーション手段に加えて，ソーシャルネットワーク Web サービスが搭乗したことにより，ユーザはより多くの他のユーザと接続しやりとりを行う傾向にある．

このような背景において，限りある “人間の注意” のリソースは、コンピューティングにおける新しいボトルネックであると指摘 [7] されている．人間ユーザの視点から見れば，過度に多い情報がもたらされる状況は，広義に「情報過多」として知られる．

本研究では特に，**割り込み過多** *(interruption overload)* 現象，言い換えれば，コンピュータシステムからの過剰量の通知やその不適切な配送による割り込みによって引き起こされる「ユーザの注意の散漫」に着目する．そのうえで本研究では，ユーザの割り込み過多を解消をめざし，ユーザの注意状態に適応的なシステム側での通知動作を実現するため，ユーザの注意を検出する仕組みを提案する．

ユーザへの様々な通知を即座に配送する，従来型の典型的な通知システムは，ユーザの生産性に負の影響を与えることが先行研究 [1], [2], [5], [17] における管理下試験で明らかになっている．本研究は，ユーザが何らかのアプリケーション操作をともなってユビキタスコンピューティングにおける目標行動を行う環境における，“Application as a Sensor” (AaaS) アプローチに基づくユーザの注意状態の検知を提案する．本システムは，ユーザの生産性を保つシステムの適応的動作実現のため，ユーザの現在の注意対象と “breakpoint”[19] を，ユーザが操作する複数のデバイス上のアプリケーションそれぞれに特化した情報を使って検知する．またその検知において，モバイルデバイス上でリ

[1] 慶應義塾大学大学院 政策・メディア研究科
Graduate School of Media and Governance, Keio University
[2] 慶應義塾大学 環境情報学部
Faculty of Environment and Information Studies, Keio University
[a)] slash@ht.sfc.keio.ac.jp

アルタイムに，生体センサを用いずに実現する点を特徴とする．これは関連する他の研究が，主に研究所内のデスクトップ環境を焦点とし，生体センサを使用して，遡及的な検知を行うのとは対照的である．

本研究の貢献は，マルチデバイス環境において，専用の生体センサを用いることなく，実時間にユーザの注意状態を検知するシステムの設計を提案する点にある．

本稿では，2 章にて通知システムによって引き起こされる割り込み過多について述べる．次に 3 章では，アプリケーション層の情報を利用したユーザの注意の検知についてその実現方針や手法について述べる．4 章では実際のシステム設計について述べる．5 章でまとめと今後の課題を述べる．

2. 割り込み過多

マルチタスク環境における過剰量の通知によるユーザへの “割り込み過多” は，限りある “人間の注意” のリソースがコンピューティングにおける新しいボトルネックであると考えられることから，情報過多に関する研究の中でも次第に注目を集めている．近年，Interruptions and Multitasking (IM) という分野において多くの研究 [10] が行われている．

割り込み過多の主な要因はコンピューティングシステムからの通知である．通知はもともと，情報をユーザの注意の焦点の外側から，より迅速にまた適時的にユーザへ提供するために設計された仕組みである．しかしながら，そういった便益にも関わらず，ユーザへの様々な通知を即座に配送する典型的な通知システムは，ユーザの生産性に負の影響を与えることが先行研究 [1], [2], [5], [17], [21] によって明らかになっている．また他の研究 [8] は，ユーザがその時遂行している主たるタスクと通知情報の間にある文脈の違いが，もうひとつの原因であることを明らかにしている．

通知は，実際のシステムにおいては，ユーザによって設定可能であり無効化することもできるが，単純に無効化する事は通知自身の便益を打ち消し，ユーザの「適時的な情報の受信」のニーズを満たせなくなることを意味する．実際に Iqbal[16] らの研究では，多くのユーザが，単純に通知システムを無効化して手動で情報をチェックするよりも，割り込みのコストを考慮に入れても通知による情報配信を好んだ．

通知がユーザによって知覚・認識されるとき，ユーザの注意リソースのうちいくらかが，通知で配送された情報へと割り当てられる．これが「割り込み」が発生する瞬間である．人間の注意は有限のリソース [20] であるので，注意のいくらかを通知へ分割して割り当てることは，“devided attention (DA)” という状態を作り出す．そしてこの DA 状態は，人間の記憶能力に大きな負の影響を与える [4] ことが知られている．従って通知が引き金となって生じる DA 状態においては，ユーザが現在行っている主要なタスク (primary task) の効率が容易に影響を受ける．

こういったユーザの割り込み過多状態を緩和するため，現在のユーザの注意状態や通知される情報に応じて動的に通知のタイミングやメディア，内容を調節する「適応的な通知サポート」が必要とされる．本研究は，適応的な通知サポートの中でも，“いつ通知を行えば良いか”，“どういった情報を通知すれば良いか” の推定のために，ユーザの注意状態の検知に焦点をあてる．

1 章で紹介した近年のユビキタスコンピューティングの流れにおける通知の特徴として，本研究では下記に焦点を充てる．

- **通知元や通知の種類の多様化:** 例として近年急速に広がりつつあるソーシャルネットワーク上の他ユーザからの情報更新や，参加型センシング [3] におけるセンシングの依頼などがあげられる．
- **緊急度合いの多様化:** 受信したユーザは数秒以内に物理的な行動を起こす必要がある緊急地震速報 [12] は，新しい通知の緊急度合いを実装する例としてあげられる．
- **終日継続する割り込み過多:** ユビキタス環境においてユーザは，常にモバイルデバイスを持ち歩き，ユビキタスサービスを日常的 (朝起きてから夜寝るまで．時には寝ている間も) に利用する．従って割り込み過多はユーザの生活に密着して終日継続する．

また本研究では，ユーザの “ユビキタスコンピューティングにおける，何らかの目的をもった対話的なタスク” の間における割り込みを対象とする．ドキュメント編集，ビデオの閲覧，e-book の閲覧，ゲームのプレイなどが例としてあげられるこの種のタスクは，情報空間の何らかのコンテンツの操作 (読み込みまたは書き込み) を含む．これは筆者らが，ユビキタスコンピューティングにおいて，不適切な割り込みの脅威にさらされるような注意が必要とされるタスクの相当部分は，上に述べたような何らかの情報空間内のコンテンツ操作に関係すると考えるからである．この対象には，オフラインでの (物理的な本を読む) 読書や料理といったタスクは含まれない．しかしながら筆者らは，本研究における問題領域の単純化の第一段階としては，このような対象の絞り込みは適切だと考える．

3. アプリケーション層の情報を使った注意の検知

“いつ通知を行えば良いか”，“どういった情報を通知すれば良いか” の推定に向けて，本章では，アプリケーション層からの情報を利用したユーザ注意状態の検知について述べる．具体的には，ユーザが操作する複数のモバイル機器上で動作するアプリケーションからの，(1) アプリケーション層での情報，および (2) 各アプリケーション固有の

情報 を利用して検知を行う点を特徴とする．

3.1 注意状態検知の実現方針

すでに述べた研究背景と研究対象にもとづき，我々は本研究における注意検知の実現方針を下記とする．

- **モバイルデバイス上で実行できること:** タスクに関係するアプリケーションの直接的な操作を行うデバイスとして，ユーザはスマートフォンやタブレットなどのモバイルデバイスを携帯し使用する．従って本システムはこれらのモバイルプラットフォームへの適合性，例えば省電力性を備える必要がある．
- **リアルタイムに検知できること:** 最終的に通知システムの逐次的な適応的動作を実現するため，注意状態の検知はリアルタイムに行える必要がある．
- **多様な通知元アプリケーション/サービスや通知の種類に対する適用性:** システムは多様な通知に対して適用でき，簡単にデプロイできる必要がある．
- **終日の利用への親和性:** ユーザの割り込み過多が終日続くため，終日にわたってユーザが利用できるシステムであることが求められる．

3.2 割り込みタイミングとしての “breakpoint”

本提案では，通知によるユーザへの割り込みを行う目標とするタイミングに関して，心理学の分野で 1970 年代に発見された概念である “breakpoint”[19] を利用する．

breakpoint の概念は，人間の認知システム内に人間のタスクを階層的な別個の「サブタスク」に分割する機能が備わっていることを意味する．この分解された隣接するタスク同士の境界のことを breakpoint と呼ぶ．例えば，ワードプロセッサを使ったドキュメント編集というタスクを例にとると，ドキュメントを新規作成する，一文を入力する，印刷ボタンを押す，保存ボタンを押す，といった操作は，粒度の差はあるものの，ドキュメント編集タスク中の「サブタスク」と考えることが出来，これらサブタスクの境界が breakpoint である．ユーザによる対話的なコンピューティングタスクにおいては，少なくとも Fine，Medium，Coarse の 3 段階の粒度の breakpoint が存在し，信頼性を追って検知できることが先行研究によって明らかになっている [15].

Happalainen[11] らによれば，生体センサを用いてリアルタイムに注意状態，特に認知負荷を計測するには，少なくとも 2 つの生体センサを組み合わせる必要がある．生体センサを日常的に終日装備するのは，ユーザにとって負担がかかる．一方本研究のアプローチは，ユーザがもともと携帯するモバイルデバイスのみを通して，やや粗粒度ながら簡単にセンシングできる，最終的には通知に最適なタイミングとして利用可能な breakpoint という指標を用いるものである．

先行研究では，通知の配送を breakpoint のタイミングまで遅らせると，ユーザの「割り込みコスト」を減らす事が出来，具体的にはユーザの主たるタスクへの復帰時間が短くなり，主観的な欲求不満度合いが低下すること [1], [13], [14] が明らかになっている．

3.3 “Application as a Sensor”(AaaS) 手法

それでは，システムはユーザ注意対象と breakpoint のタイミングを検知できるのだろうか? 本研究はその手法として，“Application as a Sensor”(AaaS) 手法を提案する．

本手法では，モバイルデバイス上でユーザによって操作されているアプリケーションに関する，(1) 開発時に内部に埋め込まれる比較的静的な情報や，(2) 実行時の状態と各種のシステム内内部イベントなどの動的な情報といった複数種類の情報を使い，breakpoint のタイミングを検知する．

本研究で対象とする，ユビキタスコンピューティングにおける何らかの目標に向かったユーザのタスクは，通常コンピュータシステム内で何らかの「アプリケーション」を伴う．したがって，ユーザが操作する対象となっているアプリケーションから可能な限りの情報を取り出し活用するこのアプローチは，理にかなっていると筆者らは考える．

4. システム設計

本章では，注意検知のためのシステムのより詳細な設計について述べる．まずはじめに，検知のために収集する情報について述べる．次に，アテンション検知の 3 つのフェーズを説明し，実行時のシステム概要について解説する．

4.1 収集する情報

表 1 に情報収集の詳細手法を示す．また表 2 に，各詳細手法における情報の入力の違いについて示す．

4.1.1 各アプリケーションに特化した情報

各アプリケーションに特化した情報のひとつの情報源として，アプリケーション開発者は明らかに一つの候補である．アプリケーション開発フェーズにおける一つの可能性は，開発者がアプリケーションのソースコード内部に，breakpoint の発生を明示的に宣言し検知システムが実行時にそれを受信するための，追加の API 呼び出しを明示的に埋め込むことである．

アプリケーション毎に特化した情報の代表的な例として，下記をあげる．

- **特定部分実行時の breakpoint 発生予測:** アプリケーション開発者は，ソースコード内の任意の場所で，アプリケーション実行時にその箇所が実行された時点で利用ユーザの breakpoint の発生が予想される場合，それを明示する API 呼び出しを設置する．例えば，アプリケーション内で画面が切り替わる場所は，そのよう

表 1 breakpoint 検知の詳細手法と利用する情報の種類

Breakpoint Detection Sub-approaches	Collected Data
Application-specific breakpoint knowledge	Explicit breakpoint declaration, Explicit future breakpoint forecast
Application run-time status/event	Stack trace, number of threads, thread names, memory consumption Android API invocation, system call invocation, rendered screen image, Low-level GUI events (tapping, keyboard typing, pinching, scrolling etc.)
System run-time status/event	starting, closing, switching applications, mode of current screen (Application, Home, Notification Center)

表 2 各詳細手法と情報の入力タイミング

Breakpoint Detection Sub-approaches	Breakpoint Knowledge Input		Data Collection at Application Run-Time
	Application Development Phase	System Training Phase	
Application-specific breakpoint knowledge	Embedding additional API calls for explicit breakpoint knowledge input (by application developer)	None	From API calls embedded inside application
Application run-time status/event	None	Tagging into collected status/event information (by application users)	From the system below application
System run-time status/event			

な API 呼び出しを設置する箇所の一候補である．

- **特定分部実行後の breakpoint 発生予測:** アプリケーション開発者は，ソースコード内の任意の場所で，アプリケーション実行時にその箇所が実行された後，特定の時間内に利用ユーザの breakpoint の発生が予想される場合，それを明示する API 呼び出しを設置する．例えば，一つのステージが最大でも 3 分で終了するゲームアプリケーションの場合，ステージが開始された時点で，アプリケーションは「3 分以内に breakpoint が発生する」と予測する事ができる．
- **注意の対象:** アプリケーション実行時の特定のデータ (例えば開いているファイル等) に関して，アプリケーション開発者は，それがユーザの現在の注意対象の推定に役立つデータであることを，API 呼び出しを通じて明示する．

4.1.2 ランタイムのシステムおよびアプリケーション状態

しかしながら，前述のような各アプリケーションに特化した情報の利用のみでは，アプリケーション開発者によるアプリケーションソースコードの改修，再コンパイル，配布などの手間が必要となる．従って，ユーザがモバイルデバイス上で多くのデベロッパーからのアプリケーションを利用する現実的な利用環境においては，この方式のみに依存するのは現実的でない．そこで本システムは，これらに加えて，システム (およびアプリケーション) の実行時に，表 1 で示した各種のアプリケーションやシステムの状態を積極的に利用する．

システムの学習フェーズにおいては，これらの状態情報のストリームがシステム内で収集され，breakpoint の発生に関するユーザからの明示的なアノテーションと共に，「教師データ」として保存される．この教師データは機械学習エンジン (Weka[18]) に入力され，breakpoint 発生検知のモデル作りに活用される．

学習フェーズは，モバイルプラットフォーム上で開発，配布されている数多くのアプリケーションをサポートするために，多くのユーザの協力を必要とする．我々はこのフェーズにおける大規模な教師データの収集とモデル作りを目的として，分散的な学習フレームワークを開発している．本フレームワークにおいては，教師データ収集に協力するユーザは，Android の Google Play ストアから我々のシステムをダウンロードし，ユーザの手元のデバイスで動作している任意のアプリケーションに関して教師データの収集・作成を行う事ができる．教師データは中央的なサーバへアップロードされ，収集されたデータを元に，サーバ上で一括してモデル作りが行われる．

4.2 システム概要

図 1 に，単一デバイス上での，本システムの実行時のシステム概要図を示す．また図 2 に，ユーザの複数デバイスを横断したシステム概観図を示す．

実行時フェーズにおいては，各アプリケーションに特化した情報，システム (OS) レベルの各種 I/O イベント (スクリーンタップやキーボードタイピング等) ，アプリケーションレベルのイベント (GUI 関連イベント等) が，breakpoint Detector へ送信される．前述の学習フェーズで作られたモデルを使った機械学習の分類器が，それらの入力を元に breakpoint の検出を行う．分類器からの検

出結果は，アプリケーション内部に埋め込まれた API 呼び出しからの breakpoint 発生予測情報とともに処理され，最終的な breakpoint 検出が行われる．ユーザが所持する複数のモバイルデバイスに横断して，ネットワークを介して breakpoint 検知の情報は共有され，デバイス横断的な breakpoint 検出が実行される．

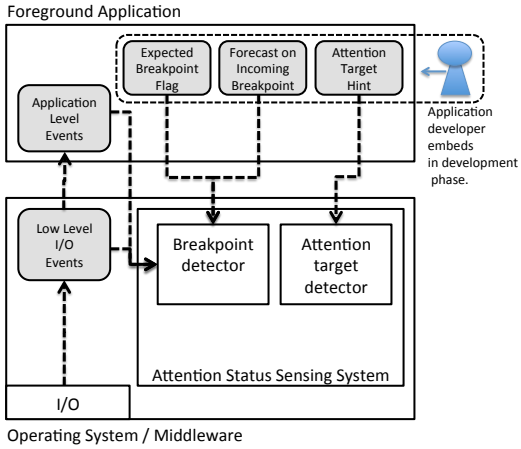

図 1 単一デバイス上での本システムのシステム設計

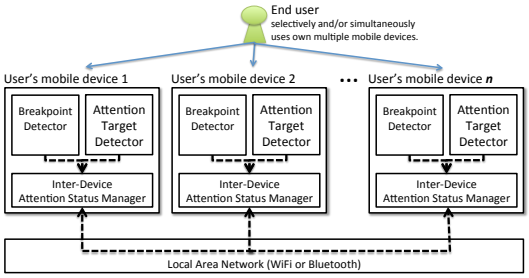

図 2 複数デバイス上での本システムのシステム設計

5. まとめと今後の課題

本稿では，コンピューティングにおける新しいボトルネックであるユーザの “注意” に適応的な通知動作を実現するための，注意状態を検知する新しい手法 Application as a Sensor (AaaS) を提案した．本手法は，モバイルデバイス上で，実時間で，生体センサを用いることなく，割り込みのタイミングの有力な候補となる breakpoint を検知する．現在，本設計に基づいたプロトタイプ実装を完了し，定性的および定量的な性能評価を実施している．

現在システムのプロトタイプ実装にもとづく評価を行っている．本システムが Android の Google Play ストアを使って広く配布可能であることを利用し，多くのユーザからの，多様なアプリケーションを対象とした教師データの収集を含む評価が今後の課題である．

参考文献

[1] Adamczyk, P. D. and Bailey, B. P.: If not now, when?: the effects of interruption at different moments within task execution, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, pp. 271–278 (2004).

[2] Bailey, B. P. and Konstan, J. A.: On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state, *Computers in Human Behavior*, Vol. 22, No. 4, pp. 685 – 708 (2006).

[3] Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S. and Srivastava, M. B.: Participatory sensing, *In: Workshop on World-Sensor-Web (WSW 2006): Mobile Device Centric Sensor Networks and Applications*, pp. 117–134 (2006).

[4] Craik, F. I., Govoni, R., Naveh-Benjamin, M., Anderson, N. D. et al.: The effects of divided attention on encoding and retrieval processes in human memory, *Journal of Experimental Psychology-General*, Vol. 125, No. 2, pp. 159–179 (1996).

[5] Czerwinski, M., Cutrell, E. and Horvitz, E.: Instant messaging: Effects of relevance and timing, *People and computers XIV: Proceedings of HCI*, Vol. 2, British Computer Society, pp. 71–76 (2000).

[6] Deloitte Touche Tohmatsu Limited: Deloitte Global Mobile Consumer Survey 2012 (2012).

[7] Garlan, D., Siewiorek, D., Smailagic, A. and Steenkiste, P.: Project Aura: toward distraction-free pervasive computing, *Pervasive Computing, IEEE*, Vol. 1, No. 2, pp. 22 –31 (online), DOI: 10.1109/MPRV.2002.1012334 (2002).

[8] Gillie, T. and Broadbent, D.: What makes interruptions disruptive? A study of length, similarity, and complexity, *Psychological Research*, Vol. 50, No. 4, pp. 243–250 (1989).

[9] Google Inc.: The New Multi-Screen World — Think with Google (2012).

[10] Gould, S., Brumby, D., Cox, A., González, V., Salvucci, D. and Taatgen, N.: Multitasking and interruptions: a SIG on bridging the gap between research on the micro and macro worlds, *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, ACM, pp. 1189–1192 (2012).

[11] Haapalainen, E., Kim, S., Forlizzi, J. F. and Dey, A. K.: Psycho-physiological measures for assessing cognitive load, *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 301–310 (2010).

[12] Hoshiba, M., Kamigaichi, O., Saito, M., Tsukada, S. and Hamada, N.: Earthquake early warning starts nationwide in Japan, *Eos, Transactions American Geophysical Union*, Vol. 89, No. 8, pp. 73–74 (2008).

[13] Iqbal, S. T. and Bailey, B.: Leveraging characteristics of task structure to predict the cost of interruption, *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 741–750 (2006).

[14] Iqbal, S. T. and Bailey, B. P.: Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption, *CHI'05 extended abstracts on Human factors in computing systems*, ACM, pp. 1489–1492 (2005).

[15] Iqbal, S. T. and Bailey, B. P.: Understanding and developing models for detecting and differentiating breakpoints during interactive tasks, *Proceedings of the SIGCHI conference on Human factors in computing*

systems, ACM, pp. 697–706 (2007).
[16] Iqbal, S. T. and Horvitz, E.: Notifications and awareness: a field study of alert usage and preferences, *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pp. 27–30 (2010).
[17] Kreifeldt, J. G. and McCarthy, M. E.: Interruption as a test of the user-computer interface, *JPL Proceeding of the 17 th Annual Conference on Manual Control*, pp. 655–667 (1981).
[18] Machine Learning Group at the University of Waikato: *Weka 3: Data Mining Software in Java*.
[19] Newtson, D. and Engquist, G.: The perceptual organization of ongoing behavior, *Journal of Experimental Social Psychology*, Vol. 12, No. 5, pp. 436–450 (1976).
[20] Simon, H. A.: Designing organizations for an information-rich world, *International Library of Critical Writings in Economics*, Vol. 70, pp. 187–202 (1996).
[21] Zijlstra, F. R., Roe, R. A., Leonora, A. B. and Krediet, I.: Temporal factors in mental work: Effects of interrupted activities, *Journal of Occupational and Organizational Psychology*, Vol. 72, No. 2, pp. 163–185 (1999).