Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

Inboxによる文書整理システムの検討

池田 ゆう子　乃村 能成

概要：我々は，様々なアプリケーションを利用しており，各アプリケーションで利用した情報は作業という観点で関連している．この関連を管理するために，本稿では Inbox による情報整理手法を提案した．提案手法では，各アプリケーションで利用した情報を収集する．そして，それらを統一的に表現することで Inbox に一覧表示する．ユーザは，表示された項目を作業に関して整理できる．アプリケーションで利用した情報をすべて収集したとき，Inbox に表示される情報量が膨大になる．この増大を抑えるために，文書のファイルパス名に着目した整理支援の仕組みを提案した．具体的には，文書がもともとフォルダ分けにより管理されていることに着目し，作業を最も表わすフォルダを推定し，ユーザに提示する．ユーザは，提示されたフォルダを作業に関して整理するため，文書を１つ１つ整理する手間を削減できる．

キーワード：文書管理，情報整理，作業再開の支援

1. はじめに

我々は，作業をするときに様々なアプリケーションを利用している．たとえば，ドキュメントを編集するときはエディタを利用し，参考資料を探すときは Web ブラウザを利用し，他者と会話するときはチャットツールを利用する．これらの情報はアプリケーションをまたがって，作業という観点で関連している．この関連は，中断した作業を再開したり，過去の作業と似た作業を行うときに有用である．たとえば，中断した作業を再開する場合は，まず中断前に参照していた文書を開き直す必要がある．あるいは，過去に参考にしていた Web ページがあるとすれば，作業再開後も同じ Web ページを参考にする確率は高い．作業開始時に，参照していた情報を想起することは手間であり，もし，過去の作業中に参照していた情報が何らかの形で管理されていれば，作業を開始するときのコストを削減できる．しかし，各アプリケーションで利用した情報は，アプリケーション毎に管理されており，作業に関しての関連が明示的に示されていない．たとえば，ドキュメントはフォルダを用いて整理し，Web ページはブックマークを用いて整理するが，あるドキュメントとあるブックマークが１つの作業に関連していることは明示的に示されていない．よって，我々はこの関連を頭の中で管理している．このような頭の中で情報を管理する体系は壊れやすく，アプリケーション毎に管理される情報の関連を有効活用できていない．統一的な情報整理は，たとえば医療現場における診療記録についても同様の問題が発生しており，診療記録を診療科によらず統一的に整理するシステムが提案されている [1]．

我々は，統一的な情報整理の手法として，Inbox による情報整理手法を提案する．Inbox とは，TODO 管理でよく用いられる GTD 方式 [2] をアプリケーション情報に適用したものである．Inbox は，各アプリケーションの属性とは関係なく，参照時刻順に情報を一覧表示し，それらを統一的に整理可能なユーザインタフェースである．Inbox を実現することで，各アプリケーションで利用した情報を作業という観点で統一的に整理できると考える．

しかし，各アプリケーションで利用した情報を Inbox にすべて一覧表示すると，Inbox 内の情報量は膨大になる．このため，メーラのメール自動分類のような情報整理を支援する仕組みが必要である．そこで我々は，日頃参照することの多い，自身の計算機内にファイルとして存在する文書を整理する手法について考える．具体的には，文書をフォルダ分けにより管理する習慣をもっていることに着目する．すると，フォルダの中には，作業の階層を表わすフォルダや作業と強く関連するフォルダが存在する．このようなユーザが行った作業を最も表わすフォルダをワーキングディレクトリと呼ぶ．整理時に Inbox に現れた文書を１つ１つ整理するのではなく，ワーキングディレクトリを作業と関連づけることで，ワーキングディレクトリ以下の文書を一括で整理できる．つまり，ユーザが参照・編集した文書一覧から，ワーキングディレクトリを推定・提示す

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

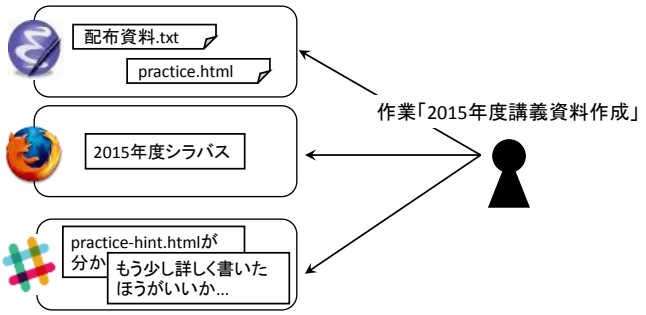

図 1 複数のアプリケーションの利用例

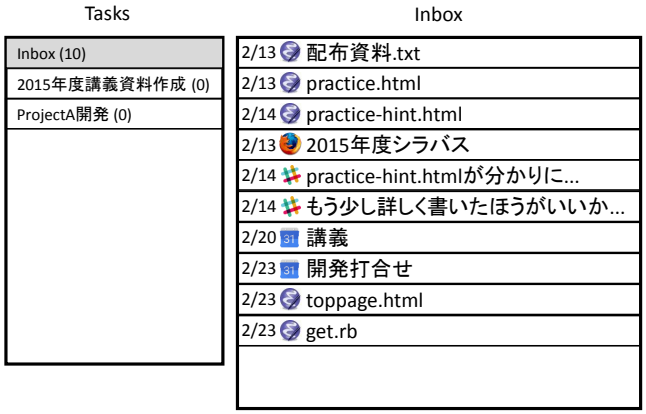

図 2 Inbox でのアプリケーション情報の表示

ることで，Inbox による文書整理を支援できると考える．

本稿では，Inbox による文書整理システムについて検討する．具体的には，まず，アプリケーション情報の整理の現状について述べ，Inbox による情報整理手法を提案する．次に，文書整理の支援手法として，文書が保存されているフォルダの中にワーキングディレクトリが存在することに着目し，ワーキングディレクトリを推定・提示する手法について述べる．

2. Inbox による情報整理

2.1 現状

1 つの作業においていくつかのアプリケーションを利用することは珍しくなく，これらのアプリケーションで利用した情報は作業という観点で関連している．たとえば，「2015 年度講義資料作成」を行うことを考える．このとき，ユーザは**図 1** に示すように，エディタで「配布資料.txt」や「practice.html」を作成し，ウェブブラウザで「2015 年度シラバス」を参照し，チャットツールで TA と相談する．そして，アプリケーションで利用したこれらの情報は，「2015 年講義資料作成」という作業に関連している．この関連は，作業を再開するときや過去の作業と似た作業を行うときに有用である．たとえば，「2016 年度講義資料作成」を行うときは，「2015 年度講義資料作成」を行ったときに参照・編集した「配布資料.txt」や「2015 年度シラバス」などを参照することで，作業の全体像や，作成すべきドキュメント，参考資料，および作業時に気をつけるべきことを把握できる．このように，いくつかのアプリケーションで利用した情報は，作業という観点で関連しており，この関連は作業の再開や過去の作業と似た作業を行うときに有用である．

しかし，各アプリケーションで利用した情報は，アプリケーション毎に管理されており，アプリケーションをまたがった作業に関しての関連が明示的に示されていない．このため，「2016 年度講義資料作成」を行うときは，エディタやウェブブラウザなどの各アプリケーションを開いて「2015 年度講義資料作成」を行ったときに利用した情報を探す必要がある．

2.2 方針

前節で述べたように，アプリケーション毎に管理されている情報は作業という観点で関連しているが，我々はこの関連を明示的に管理していない．そこで，我々は各アプリケーションで利用した情報を明示的に整理・管理する手法として Inbox による情報整理手法を提案する．本手法の方針は以下の通りである．

(方針 1) 各アプリケーションで利用した情報を Inbox に収集し，参照時刻順で表示

(方針 2) Inbox に表示された情報を作業に関して整理

Inbox とは，タスク (TODO) 管理の手法の 1 つである GTD において，TODO を収集する場所のことである．GTD では，TODO を収集し，収集した TODO をレビューすることで整理を行う．レビューとは，収集した TODO を見直し，適切に分類することである．我々はこれまで，GTD の情報整理手法をカレンダ情報に適用し，Inbox によるカレンダ情報の整理手法を提案した．また，その有用性を示した [3]．カレンダ情報の整理においての Inbox は，予定の情報を時系列とは異なる軸で整理可能なメールアプリケーションの受信箱に似たユーザインタフェースである．カレンダ情報の整理に Inbox を利用することで，未整理の予定の数を正確に把握でき，整理効率を上げられる．

この GTD の情報整理手法と Inbox をアプリケーション情報全般の整理に適用する．Inbox による情報整理手法については次節で詳しく述べる．

2.3 Inbox による情報整理手法

Inbox による情報整理手法では，作業中に各アプリケーションで利用した情報をすべて Inbox を用いて整理する．具体的には，まず，各アプリケーションで利用した情報の所在や更新時刻といったメタ情報を自動で収集し，統一的な形式で保存する．次に，「いつ情報を参照したか」という時刻で情報をソートし，Inbox に一覧表示する．そして，ユーザは自身が行った作業を表わす仮想的なフォルダ (Task) を作成し，Inbox に表示された情報を Task に関し

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

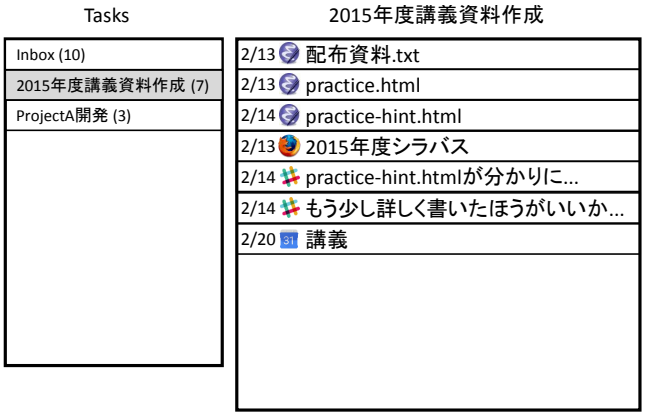

図 3 情報整理後の Inbox

て整理する．**図 2** に情報収集後の Inbox のイメージを示す．Inbox には，ユーザが各アプリケーションで利用した情報が一覧表示されている．たとえば，エディタで作成した「配布資料.txt」とウェブブラウザで閲覧した「2015 年度シラバス」といった，異なるアプリケーションで利用した情報を Inbox を用いることで一度に確認できる．そして，ユーザは「2015 年度講義資料作成」という Task を作成し，この Task に関する項目を整理する．整理後には，Task を選択することで関連する全ての情報を一覧できる．**図 3** に整理後の Inbox のイメージを示す．Task「2015 年度講義資料作成」には関連する情報が 7 個存在することが確認できる．また，関連する情報がエディタ，ウェブブラウザ，チャットツーツ，およびカレンダに存在することがわかる．よって，「2016 年度講義資料作成」を行うときは，各アプリケーションを開き情報を探す必要がなく，Inbox の Task「2015 年度講義資料作成」を確認するだけで，各アプリケーションで利用した情報を把握できる．

3. 整理の支援

3.1 概要

2 章で述べた Inbox による情報整理手法の実用化を考えたとき，日々利用した情報を複数のアプリケーションから収集するため，Inbox に収集される項目が多くなると考えられる．よって，すべての項目を手作業で整理することは手間となる．そこで，情報を自動分類したり，分類先をサジェストしたりする仕組みが必要であると考える．たとえば，メーラは，ユーザが事前にルールを定義しておくことでルールに従いメールをフォルダに自動分類する仕組みがある．さらに，メールの類似度を算出することで，分類ルールの生成を行うシステムも存在する [4]．このような整理の支援により，メールをフォルダに分類する作業のコストは圧倒的に削減されているといえる．また，写真の整理を例にとると， 撮影時間の密度の高い写真群をグループ化し，自動でアルバムを生成するシステムも存在する [5]．このような，情報の整理を支援する仕組みが Inbox にも必要である．

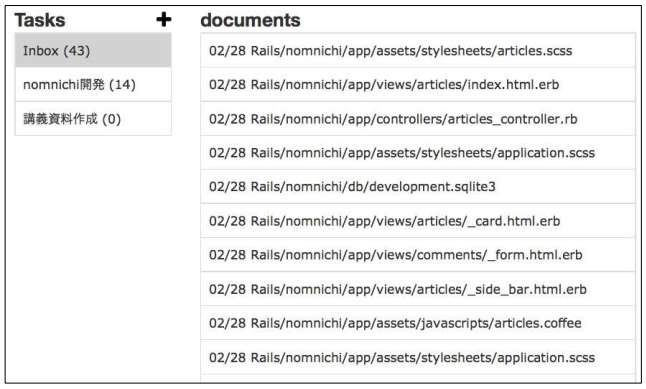

図 4 文書収集後の Inbox

3.2 文書整理の支援

本稿では，アプリケーション情報の中から，特に文書整理の支援について述べる．

Inbox を用いて文書整理を行うことを考える．まず，ユーザは自身の計算機内の適当な場所に文書をファイルとして作成し，作業中に参照する．その後，システムは，参照した文書を自動的に収集し，Inbox にそのファイルパスを表示する．そして，ユーザは，Inbox に表示された項目を Task に整理する．この手法で文書を整理することを考えると，Inbox には参照したすべての文書が一覧される (**図 4**)．図 4 のように参照した文書がすべて一覧表示されると，Inbox 内は繁雑になり，表示されている文書を 1 つ 1 つ整理することは非常に手間である．

文書整理の問題においては，参照や編集といったユーザのファイル操作履歴から関連性を求める手法 [6] やひとつの作業に関連するファイルは頻繁に近い時間に参照されるという性質を利用したファイルのクラスタリング手法 [7] が存在する．

我々は，文書がもともとファイルとしてフォルダ分けによりある程度整理されていることに着目する．参照した文書はいずれかのフォルダに属していることが多い．このため，Inbox に文書をすべて表示するのではなく，フォルダを提示し，ユーザはフォルダ単位で文書を整理することで整理の手間を削減できる．たとえば，あるユーザは，**図 5** のようなフォルダ分けで「図 1.pdf」，「文書 1.txt」，「文書 2.txt」，および「シート 1」を管理している．ある作業中にこの 4 つの文書を参照すると，Inbox には 4 つの文書が収集される．そして整理時にこの文書を 1 つ 1 つ Task に整理するのではなく，たとえば，図 5 中の「資料」フォルダを Task「講義資料作成」と結びつける．そうすれば，そのフォルダ以下の文書である「図 1.pdf」と「文書 1.txt」は，明示的に整理しなくとも，Task「講義資料作成」と関連することは明らかである．

ここで重要なことは，どのフォルダをユーザに提示し，

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

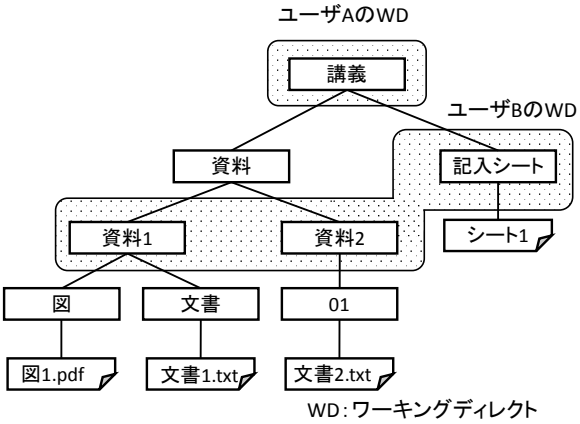

図 5 編集した文書

どのフォルダを Task と結びつけるかである．文書が属するフォルダをすべて提示すると，かえって整理対象の項目が多くなる．図 5 の例の場合，文書ファイル 4 個に対して，8 個のフォルダが整理対象となる．このため，文書をフォルダ単位でユーザに提示するといっても，どのフォルダをユーザに提示するかが重要である．ここで，フォルダの中には特定の作業と関連しているフォルダが存在する場合がある．図 5 では，たとえば，「資料」フォルダや「記入シート」フォルダがそれに相当し，それぞれ「講義資料作成」や「記入シート作成」という作業と関連している．このように，ユーザが行った作業を最も表わすフォルダをワーキングディレクトリと呼ぶ．このワーキングディレクトリを推定し，ユーザに提示できれば，整理の工数を大幅に削減できると考える．そこで，ワーキングディレクトリを推定する手法について次章で詳しく述べる．

4. ワーキングディレクトリの推定

4.1 ワーキングディレクトリとは

ワーキングディレクトリとは，作業を最も表わすフォルダである．この「作業を最も表わすフォルダ」とは何なのかを機械的に決定することは難しい．なぜならば，「作業を最も表わすフォルダ」とは，ユーザが作業をどの単位で意識するかによって異なるからである．たとえば，3.2 節で述べた図 5 のフォルダ構成の下，ユーザ A とユーザ B が作業中に「図 1.pdf」，「文書 1.txt」，「文書 2.txt」，および「シート 1」を参照したとする．また，ユーザ A とユーザ B が意識する作業はそれぞれ以下の通りである．

(1) ユーザ A が意識する作業
　(a) 講義準備
(2) ユーザ B が意識する作業
　(a) 資料 1 作成
　(b) 資料 2 作成
　(c) 記入シート作成

この場合，ユーザ A にとってのワーキングディレクトリは図 5 中に示した「講義」のみであり，ユーザ B とってのワーキングディレクトリは，図 5 中に示した「資料 1」，「資料 2」，および「記入シート」の 3 つである．つまり，ワーキングディレクトリとはユーザが意識する作業によって異なる主観的な存在である．

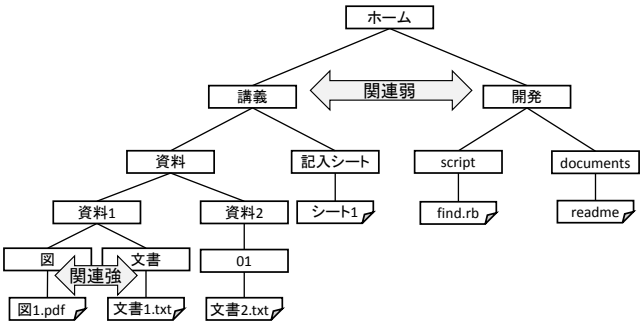

図 6 フォルダ間の関連

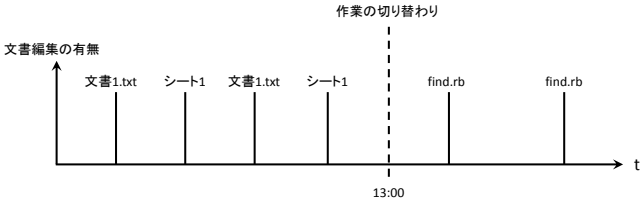

図 7 タイムラインの例

4.2 ワーキングディレクトリの推定手法

4.2.1 方針

3.2 節で述べたように，Inbox による文書整理を支援するには，文書 1 つ 1 つを整理するのではなく，作業を最も表わすワーキングディレクトリという単位で文書を整理することが効果的である．このため，収集した文書を Inbox に一覧表示する際にワーキングディレクトリを推定し，ユーザに提示する手法を提案する．

編集した文書群を作業ごとに分類し，作業ごとの文書のファイルパスを比較すると，上層のフォルダの違いにより，文書をある程度作業に分類できるとわかった．これは，フォルダ構造において，階層が上層になるほどフォルダ間の関連が弱くなるからである．たとえば，図 5 に「開発」のフォルダを加えた**図 6** のようなフォルダ構成を例に考える．図 6 の場合，「図 1.pdf」と「文書 1.txt」は同一の作業に関連しており，各文書を包含する「図」と「文書」の関連は強い．一方で，「図 1.pdf」と「find.rb」については，「講義」と「開発」の関連が弱く，2 つの文書が 1 つの作業に関連しているとは考えがたい．ここで，作業中に文書を保存したタイミングを文書の編集時刻とし，1 日に編集した文書の編集時刻を時刻順に並べたものをタイムラインと呼ぶ．タイムラインを観察すると，隣り合う編集時刻の文書のファイルパスで上層のフォルダが切り替わっている部分が存在した．図 6 のフォルダ構成の下で作業を行ったユーザのタイムラインを**図 7** に示す．図 7 の 13 時に注

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

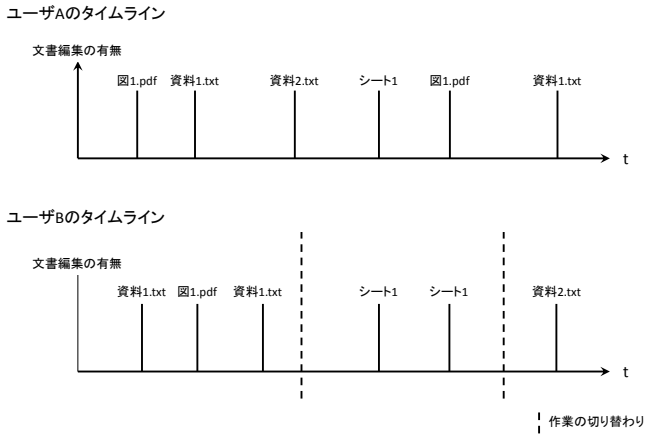

図 8 ユーザ間のタイムラインの違い

目し，「シート 1」と「find.rb」のファイルパスを比較する．すると，フォルダが「講義」から「開発」に切り替わっていることがわかる．この部分が作業の切り替わりの時刻だと考えられる．これらのことから，タイムライン上で上層のフォルダが切り替わった部分を作業の切り変わりと判定し，その地点でタイムラインを分割する．そして，分割されたタイムライン上の文書を包含するフォルダをワーキングディレクトリと推定する．つまり，具体的な手順としては，まず，以下の手順を行う．

(手順 1) 上層のフォルダが分かれる部分でタイムラインを分割

(手順 1) のみを行ってワーキングディレクトリを推定すると，ワーキングディレクトリが上層のフォルダに集中する．たとえば，図 6 のフォルダ構成に (手順 1) を適用すると「講義」と「開発」がワーキングディレクトリと推定される可能性が高い．これは，ユーザ A のように意識する作業の粒度が大きいユーザにとっては適切なワーキングディレクトリといえる．しかし，意識する作業の粒度が小さいユーザにとっては，上層のフォルダがワーキングディレクトリとなることは好ましくない．たとえば，ユーザ B のように「資料 1 作成」，「資料 2 作成」，および「記入シート作成」といった小さい粒度の作業を意識しているユーザにとっては，図 6 において，ワーキングディレクトリは「講義」ではなく，「資料 1」，「資料 2」，および「記入シート」である．

そこで，編集時刻の交錯を用いることで，より正確なワーキングディレクトリを推定できると考える．タイムラインを観察すると，同じ作業に関する文書の編集時刻は交錯していることが分かった．**図 8** にユーザ A とユーザ B のタイムラインの違いを示す．ユーザ A は，作業として「講義準備」を行ったと意識しているため，図 8 上のタイムラインのように「講義」フォルダ以下の文書である「図 1.pdf」，「資料 1.txt」，「資料 2.txt」，および「シート 1」の編集時刻は交錯する可能性が高い．一方，ユーザ B は，作業として「資料 1 作成」，「資料 2 作成」，および「記入シート作成」を行ったと意識しているため，ユーザ A と異なり，「講義」フォルダ以下のすべての文書の編集時刻が交錯するわけではない．ユーザ B の場合は，「資料 1」，「資料 2」，および「記入シート」の 3 つのフォルダ以下の文書の編集時刻がそれぞれ交錯する．たとえば，「資料 1.txt」と「図 1.pdf」は同じ作業に関連した文書であるため編集時刻は交錯する．しかし，「資料 1.txt」と「シート 1」は違う作業に関連した文書であるため編集時刻は交錯しない．このように，ひとつの作業中に参照した複数の文書は編集時刻が交錯する．このことから，タイムライン上で編集時刻が交錯している文書をひとつのまとまりと考え，それらをすべて包含するフォルダをワーキングディレクトリと推定できる．つまり，(手順 1) でおおまかに分割したタイムラインを，編集時刻の交錯の有無によってさらに分割する．具体的な手順としては，(手順 1) で分割したタイムラインに対して，以下の手順を適用する．

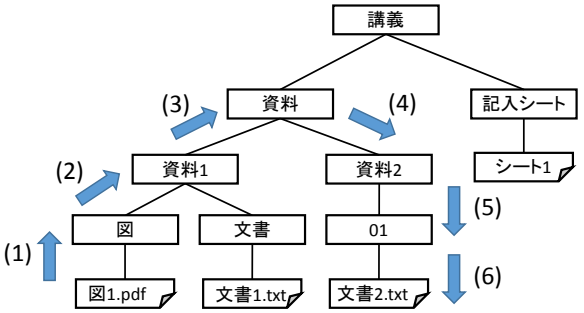

図 9 ファイルパス間の移動コスト

(手順 2) 編集時刻の交錯を用いてタイムラインを分割

これら (手順 1) と (手順 2) によって分割されたタイムライン上の文書群を包含するフォルダをワーキングディレクトリと推定する．次項以降でそれぞれの手順について詳しく述べる．

4.2.2 上層のフォルダが分かれる部分でタイムラインを分割

タイムライン上で上層のフォルダが切り変わった部分を判定するために，タイムライン上で隣り合う編集時刻の互いのファイルパスを比較する．そして，ファイルパスの相違度を算出し，相違度が一定の閾値を越えた部分を上層のフォルダが切り変わった部分と判定する．ファイルパスの相違度は以下の 2 つの観点から算出する．

(1) ファイルパス間の移動コスト

ある文書から別の文書に移動することを考えると，まず，移動前の文書から互いの共通フォルダまで遡り，次にそこから目的の文書まで辿る必要がある．このときのフォルダを移動した工数をファイルパス間の移動コストとする．**図 9** に例を示す．この場合，「図 1.pdf」から「文書 2.txt」に移動するには，図 9 中の (1) から (6) の道筋を辿る必要があり，移動コストは 6 となる．

(2) ファイルパスの深さによる重み付け

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

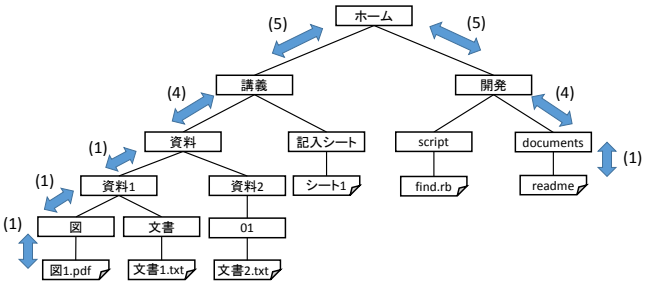

図 10　ファイルパスの深さによる重み付け

(1) の方法を用いたとき，フォルダ構造の下層部分での移動コストと上層部分での移動コストが一致したときに，両者は平等に扱われる．しかし実際は，フォルダ分けは上層になるにつれて，フォルダ間の関連は弱くなる．つまり，上層部分で移動した方がファイルパス間の相違度は大きくなるべきである．このため，ファイルパスの深さを (1) の移動コストに重み付けとして与える．**図 10** に例を示す．この場合，「ホームから開発」や「ホームから講義」といったトップフォルダを介した移動は重み 5 を付与する．また，「開発から documents」や「講義から資料」といったトップフォルダの 1 つ下層のフォルダを介した移動は重み 4 を付与する．それ以外の移動は重み 1 を付与する．

以上の観点からファイルパスの相違度を算出する．相違度が一定の閾値を越えた部分を作業の切り替わりとし，タイムラインを分割する．また，例に示した重み付けの値 (5, 4，および 1) や相違度の閾値は，個人の作業パターンからプロファイルとして与える．

4.2.3　編集時刻の交錯を用いてタイムラインを分割

(手順 1) で分割されたそれぞれのタイムラインに対して，以下のアルゴリズムを適用し，タイムラインを分割する．

(1) 最初に編集した文書の最終編集時刻をタイムラインの分割地点とする

図 11 にタイムライン分割の例を示す．最初に編集した「文書 A」の最終編集時刻である 14 時を分割地点とする．

(2) 分割地点までに編集したすべての文書に関して

(a) 文書の最終編集時刻が分割地点を越えている場合，そこを新たな分割地点とする

図 11 では，「文書 C」と「文書 B」が対象である．「文書 B」の最終編集時刻が 14 時を越えているため，14 時 10 分をあたらな分割地点とする．

(3) 分割されたタイムラインに関して (1) と (2) を適用する

4.3　調査

4.3.1　調査方法

4.2 節で述べた手法を用いて，あるユーザが編集した文書からワーキングディレクトリを推定することを試みた．

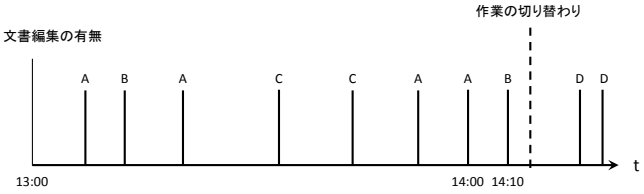

図 11　編集時刻の交錯に寄るタイムラインの分割

表 1　作業とワーキングディレクトリ (2016 年 2 月 21 日)

通番	ワーキングディレクトリ (候補)	作業名 (Task)	ワーキングディレクトリ (確定)
1	Document/meeting/192/	第 192 回打合せ資料	Document/meeting/192/
2	Ruby/documents-collector/	文書解析	Ruby/documents-collector/
3	Rails/Inbox/	Inbox 実装	Rails/Inbox/
4	GeekGirls/test/	GeekGirls で Git 勉強会	GeekGirls/
5	GeekGirls/sandbox/		
6	Documents/home/meeting/192/		
7	Dropbox/資料用/192/		
8	todo/		
9	Rails/Inbox/log/		
10	Rails/Inbox/db/		
11	Rails/Inbox/tmp/pids/		

ユーザは主な仕事として，日々，文書作成と Web アプリケーション開発を行っている．約 1 カ月間 (2016 年 2 月 18 日から 3 月 15 日) ユーザの文書更新を追跡しログを取得した．取得したログから 1 日毎にその日のワーキングディレクトリを推定する．また，ユーザには作業中に自身が行った作業を記録してもらった．ユーザは，記録した作業から自身が思うワーキングディレクトリを挙げてもらう．そのワーキングディレクトリと推定したワーキングディレクトリを比較する．

4.3.2　結果

表 1 に，2016 年 2 月 21 日において候補として提示したワーキングディレクトリ，ユーザが記録した作業，およびユーザが選んだワーキングディレクトリを示す．

編集した文書が 73 個存在し，その中から 11 個のフォルダがワーキングディレクトリと推定された．よって，Inbox に表示される情報量は 6 分の 1 に削減できたことが分かる．また，4 つの Task の内 3 つについては，候補として表示されたワーキングディレクトリとユーザの選択が一致した．しかし，通番 4 の Task「GeekGirls で Git 勉強会」のワーキングディレクトリは，表示されたワーキングディレクトリからは選ばれなかった．推定されたワーキングディレクトリ「GeekGirls/test/」,「GeekGirls/sandbox/」に対して，ユーザが選んだワーキングディレクトリはそれらを包含する「GeekGirls/」であった．

4.3.3　考察

本手法でワーキングディレクトリを推定する場合，以下の 2 つの場合が発生すると考えられる．

(場合 1)　ワーキングディレクトリが不必要に分割される

(場合 2)　ワーキングディレクトリが不必要に統合される

今回の調査では，前項で述べた Task「GeekGirls で Git 勉強会」が (場合 1) に相当する．また，(場合 2) は発生しなかった．これは，調査対象のユーザが意識する作業の粒度が大きかったからである．作業の粒度を細かく意識する

Vol.2016-DPS-167 No.30
Vol.2016-MBL-79 No.30
Vol.2016-ITS-65 No.30
2016/5/27

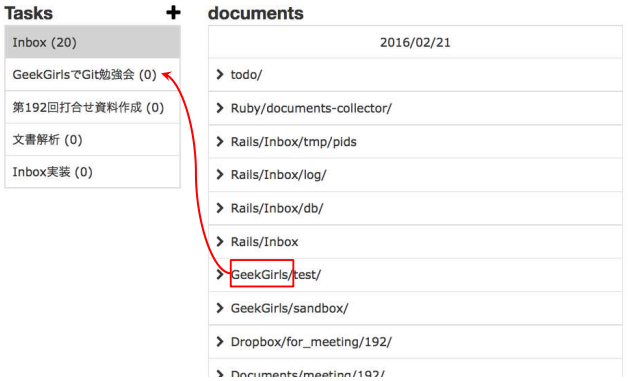

図 12 ユーザインタフェースによるワーキングディレクトリの修正

ユーザは，(場合 2) が発生しやすいと考える．たとえば，通番 3 の「Inbox 実装」について，「Rails/Inbox/」以下にスクリプトを整理するフォルダや HTML を整理するフォルダが存在する．このため，「Inbox のスクリプト作成」や「Inbox のビュー作成」といったように，細かい粒度の作業を意識するユーザからすれば，「Rails/Inbox/」がワーキングディレクトリの候補として提示されることは好ましくない．

このような誤った推定を修正するために，ワーキングディレクトリを修正するユーザインタフェースを提供する．提供するユーザインタフェースとワーキングディレクトリの修正については次節で詳しく述べる．

4.4 ユーザインタフェースによるワーキングディレクトリの修正

ユーザは，システムが提示するワーキングディレクトリの候補と自身が行った作業を比較し，Task に対応するワーキングディレクトリを指定する．このとき，ユーザは以下に示すユーザインタフェースを通じて，システムが推定するワーキングディレクトリを修正する．**図 12** にユーザインタフェースを示す．Inbox にはワーキングディレクトリ候補が一覧表示されている．ワーキングディレクトリ以下のフォルダや文書は畳み込まれて表示される．このとき，スラッシュで区切られたフォルダパスの部分をドラッグ可能である．つまり，図 12 中の「GeekGirls/test/」に対して，「GeekGirls」と「test」の両方がドラッグ可能である．また，文書そのものをドラッグすることも可能である．よってワーキングディレクトリを「GeekGirls」に修正する場合は，図 12 のように「GeekGirls」を Task にドラッグアンドドロップする．これによって，ワーキングディレクトリは「GeekGirls」に修正され，次回以降の推定ではこの操作を考慮した推定を行える．

このように，ワーキングディレクトリを修正するインタフェースを用意し，その修正を次の推定に反映することで，推定されるワーキングディレクトリは適切な個数に収束すると考えられる．

5. おわりに

本稿では，Inbox による情報整理手法を提案した．さらに，アプリケーション情報の中でも文書に着目し，Inbox による文書整理の支援方法について述べた．具体的には，ワーキングディレクトリを定義し，ワーキングディレクトリを推定・提示する手法を提案した．ワーキングディレクトリとは作業を最も表わすフォルダである．Inbox を用いて文書を整理するときに，文書そのものを整理するのではなく，ワーキングディレクトリを作業を関連づけることにより，文書整理の手間を削減できる．最後に，この手法を用いてワーキングディレクトリを推定したときの精度について調査した．調査結果から，ユーザが考えるワーキングディレクトリより多くのワーキングディレクトリが推定されることがわかった．しかし，ユーザインタフェースを通してワーキングディレクトリを修正することで徐々にワーキングディレクトリの数を適切な数に収束させることが可能である．

残された課題として，ワーキングディレクトリ推定手法の改善がある．本稿では，文書間のファイルパスの相違度を計算する際の，ファイルパスの深さによる重み付けの値と閾値は特定のユーザに対して固定値を与えた．この重み付けと閾値を算出する手法を検討する必要がある．また，他のアプリケーション情報の整理支援の検討がある．

謝辞 本研究の一部は，科学研究費補助金・基盤研究 (C)(課題番号: 26330224) による研究費を得て実施した．

参考文献

[1] 倉林則之，　竹視，上田郁奈代，藤井歩美，武田理宏，松村泰志：診療記録統合管理システム（DACS）における文書の統合管理の有効性，医療情報学，Vol. 32, No. 4, pp. 197–205 (オンライン)，DOI: 10.14948/jami.32.197 (2012).
[2] Allen, D.: *Getting Things Done: The Art of Stress-Free Productivity*, Penguin Books (2002).
[3] 池田ゆう子，乃村能成，谷口秀夫：Inbox によるカレンダ情報の整理手法，情報処理学会研究報告，Vol. 2015-DPS-163, No. 15, 第 163 回マルチメディア通信と分散処理研究発表会，pp. 1–8 (2015).
[4] 佑介平岡，忠親大圃，孝行伊藤：Self Organizing Map に基づく電子メール分類ルール自動生成手法及び評価，人工知能学会全国大会論文集，Vol. 19, pp. 1–4（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/40020230602/⟩ (2005).
[5] 綾塚祐二：Album Weaver: 密度の高い写真群からアルバムを紡ぐ，第 15 回インタラクティブシステムとソフトウェアに関するワークショップ，pp. 7–12 (2007).
[6] Wu, Y., Otagiri, K., Watanabe, Y. and Yokota, H.: A file search method based on intertask relationships derived from access frequency and rmc operations on files, *Database and Expert Systems Applications*, Springer, pp. 364–378 (2011).
[7] 小田切健一，渡辺陽介，横田治夫：頻出ファイル集合のアクセス時間を考慮した仮想ディレクトリ生成手法 (2010).