L-068

Web コンテンツの安全かつ簡便な再利用を可能とする編集技術 Visual eXcart

Visual eXcart : Web Content Edit Framework for Secure and Seamless Reuse

浜田 伸一郎† 石谷 康人†
Shinichiro Hamada Yasuto Ishitani

1. はじめに

Web のコンテンツ利用やサービス利用において、あちこちの Web サイトやシステムに散在する情報を集めて比較分類したり、さらには整理・検討結果を元に情報を再利用するなどの作業がしばしば求められる。たとえば、ツーリストなどのコンシューマ向けサービスでは、行きたい観光スポットを旅行会社・旅行記などのサイトからピックアップし、希望する期間にうまく収まるよう組合せや順序を検討する必要がある。また業務システム用途では、書類作成において、企業内の各部門が蓄積するデータの中から有用なものを引用したいというニーズがある。このようにコンシューマ向け・業務向けを含め、整理・検討作業など、閲覧後のコンテンツ活用に関する編集作業が、本質的にはタスク全体の大きなウェイトを占めていることは多い。このような作業は編集工学[*1]では「編纂」として知られ、この作業支援を行うことは、知的作業を伴う様々なタスクにおいて重要である。

しかし現状としては、コンテンツ活用に関する作業はシステム化されておらず頭の中で行わねばならない、あるいは自由度の低いフォーム画面で作業することを強いられる、といったことが多い。その大きな要因としては、Web システムの情報メディアである HTML では実現困難であるということがある。HTML は、コンテンツを閲覧することを目的として設計されたメディアである。HTML では、Web ページ間のデータやり取りやコンテンツ加工といった機能がないため、Web ページや Web サイトを越えてコンテンツを選択的に収集したり、収集したコンテンツを再利用するなどのコンテンツ活用の作業を提供することはできないという問題がある。またコンテンツの有効利用という観点では、HTML ではデータを扱うことができないので、たとえば、旅行立寄り先が予定移動時間に収まるかチェックするとか、集めた商品群を購入処理にかけるといった、タスクの目的に沿った処理ができない。HTML を情報処理収集対象とする場合、文書片の貼り合わせ以上の再利用支援を行うことは基本的に難しいという問題がある。

2. 概要

Web システムを越えた横断的なコンテンツ活用を可能とするには、Web システム間でコンテンツを相互運用できる可搬性のある情報提供手段と、配布された情報を集めて使うことができる編集手段が必要である。そこで我々は、コンテンツを「ピース」と呼ばれる再利用可能な部品群として提供できる XML-P'z という XML ベースのフォーマットと、その XML-P'z で記述された Web ページを読み込んで Web ブラウザに表示し、Web 上で簡便かつ安全にコンテンツを活用できる編集エンジン Visual eXcart を開発した。

XML-P'z 文書は、"text/xmlpz"という固有の MIME タイプを持っており、Web サーバから Web ブラウザへ当 MIME タイプの文書が送信されると、Web ブラウザの中で Visual eXcart が起動され、XML-P'z 文書を利用することができるという構成になっている。図 1 は、その構成を図示したものである。

Visual eXcart は、ドラッグドロップなどを用いた分散する Web サイト上のコンテンツをシームレスに組み合わせるなどの、簡便かつ便利な編集操作環境を提供することで、ユニバーサルなコンテンツの再利用を直感的かつ効率的に行えるようにする。一方、原理的には、任意 Web サイト間でのやりとりを可能にすることから、Visual eXcart は、利用制御およびジャーナルという仕組みにより、コンテンツやアプリケーションの想定外利用や不正コンテンツ入力を防止することで安全性を確保するとともに、活用シナリオに基づいた編集作業方法を設計することを可能とする。

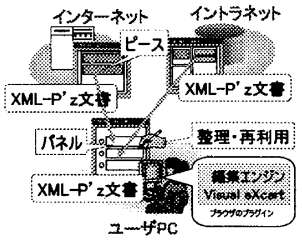

図 1: Visual eXcart のソフトウェア構成

3. 簡便なコンテンツ再利用の設計

3.1 XML-P'z 文書の定義

XML-P'z 文書を再利用可能にするには、コンテンツ部品化に関する記述を行う必要がある。そのためには、インスタンス定義とクラス定義を行う。

[インスタンス定義]

インスタンス定義の主な目的は、XML-P'z 文書内のコンテンツを用いて再利用単位であるピースを定義すること、および各ピース内部に、子ピース群を収納するコンテナであるパネルを定義することである。これにより、ピース入れ子構造を宣言的に定義すれば、再利用や整理など、後述のピース単位での基本編集機能群が Visual eXcart によって提供される。

ピースは、自コンテンツ内部に複数のパネルを持つことができるようになっており、その各パネル内に別のピース群を子供として収納できる。パネルとは、ピースを収納する名前つきコンテナである。同様に、各子ピースはパネルを通じて子ピース群を持つことができる。すなわち、XML-P'z 文書は、ピースパネルの反復再帰構造(図 2)を取る。なお文書ルートに存在するピースは特別にルートピースと呼ばれる。ルートピースは文書全体を表現するピースであり、デフォルトでは再利用できない設定になっている。

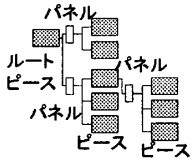

図 2: ピースパネルの反復再帰木構造

以上のピースパネル構造を設計するために必要な命令が、ピース指定子(@pz:piece)とパネル指定子(pz:panel)である。ピース指定子が付与されたエレメント以下のサブコンテンツはピースとなる。またそのサブコンテンツの任意位置において、0 個以上のパネル指定子を記述できる。このパネル指定子がそのピースが所有するパネルとなる。ただしルートピース以外のピースは、必ずパネル

†東芝研究開発センター知識メディアラボラトリー

に収納されていなければならない。したがって、ピース指定子を持つエレメントの親は、必ずパネル指定子でなければならない。

以下は商品カタログを提示するインスタンス定義例である。ルートピース下に、パネル goods があり、その下に 2 つのカタログが収納されている。ただし 2 つ目のカタログはパネル options 下にカタログを 1 つ収納している。

```
shop.pzx
<?xml version="1.0"?>
<shop pz:piece="classes.pzx#pzclass(Shop)" pz:version="1.0"
 xmlns:pz="http://www.toshiba.co.jp/xmlpz">
<pz:panel pz:name="goods">
<catalog pz:piece="classes.pzx#pzclass(Catalog)">
<name>Equiam 6010</name><os>windows</os><price>82000</price>
<pz:panel pz:name="options"/>
</catalog>
<catalog pz:piece="classes.pzx#pzclass(Catalog)">
<name>Equiam 5040</name>
<os>windows</os>
<price>79000</price>
<pz:panel pz:name="options">
 <catalog pz:piece="classes.pzx#pzclass(Catalog)">
 <name>LAN Cable 10a</name><os>windows</os><price>22000</price>
 </catalog>
 </name>
</catalog>
</pz:panel>
</shop>
```

[クラス定義]

インスタンス定義を行うだけでも、Visual eXcart によって後述の基本編集機能群が提供され、ユーザはピース単位での再利用操作が可能となるが、この状態では、ピースは単なる XML データの塊に過ぎない。ピースに対してクラス定義を割り当てることで、アプリケーションの活用シナリオに沿った付加機能を付与することができる。クラス定義は多岐にわたる種類の設定項目を持っており、これら設定項目をクラスメンバーと呼ぶ。各種クラスメンバーはすべてオプショナルである。ここでは、クラスメンバーのうちスタイル・メソッド・コンストラクタについて説明する。以下は、クラス定義の記述例である。

```
classes.pzx
<?xml version="1.0"?>
<pz:define xmlns:pz="http://www.toshiba.co.jp/xmlpz">
 <pz:class pz:name="Catalog">
  <pz:style  pz:name="normal" pz:src="n.xsl"/>
  <pz:style  pz:name="design" pz:src="d.xsl"/>
  <pz:style  pz:name="spec"   pz:src="s.xsl"/>
  <pz:style  pz:name="icon"   pz:src="s.xsl"/>
  <pz:method pz:name="buy"    pz:src="p.js#pzlogic(buy)"/>
  <pz:new    pz:src="n.xml"/>
 </pz:class>
</pz:define>
```

クラス定義を行うには、定義ブロック指定子(pz:define)とクラス定義指定子(pz:class)を用いる。定義ブロック指定子内に任意数のクラス定義を記述する。クラス定義内でスタイルやメソッドなど必要なクラスメンバーを宣言する。

スタイルは、ピースの表示方法を決定するクラスメンバーであり、クラスメンバー内に 0 個以上宣言できる。表示方法自体の定義は、pz:src 属性で指定した XSLT によって行う。スタイルがないピースは、内部 XML コンテンツがそのまま表示されるが、スタイルを持つピースは、スタイルを通じて表示される。

メソッドは、ピースに関する手続きを表すクラスメンバーであり、クラスメンバー内に 0 個以上宣言できる。ピースパネル構造を操作・参照することで文書編集を補助したり(たとえば合計値段を計算する)、ピースと紐付けのあるサービスを起動する(たとえば購入処理をかける)、などのプログラムを提供することが考えられる。ユーザは、コンテクストメニューからメソッドを呼び出すことができ、他ピースのメソッドから呼び出すことができる。プログラム自体の定義は、pz:src 属性で指定したスクリプト言語によって行う。

コンストラクタは、ピースの初期化に関するクラスメンバーであり、クラスメンバー内に最大 1 つ宣言できる。コンストラクタを持つピースは、編集中に、コンテクストメニューの新規作成コマンドなどによって新規に作成することが可能になる。pz:src 属性で指定した XML 文書で新規作成直後の初期コンテンツを定義する。

インスタンス定義によって提供される各ピースに対して、以上のクラス定義によって提供されたクラスを割り当てるには、前述のピース指定子を用いる。ピース指定子の属性値に、クラス URI という特別な URI を付与することで、ピースが所属すべきクラスを指定できる。ただしピースは必ずしもクラスに属さなくても良い。

3.2Visual eXcart の動作

[基本編集機能群]

XML-P'z 文書をブラウザで開くと、(各ピースに 1 つ以上のスタイルが定義されている場合)HTML と同じ Look & Feel で表示されるが、それに加えて、ピースとして定義されているコンテンツ部品を独立的に操作できるようになっている(図 7 参照)。

インスタンス定義によりピースの入れ子構造が定義されていれば、ユーザは、XML-P'z ページ内・ページ間を問わず、任意のパネルから任意のパネルへピースをドラッグ＆ドロップによって移動・コピーできる。HTML では、ページ内での操作しかできず、ページ間でのコンテンツ連携はできないが、XML-P'z 文書は、ピースを通じてページ間でデータをやり取りできる。Web サイト内での利用においては、整理・比較分類などユーザの検討作業への適用が有効である。HTML では、単一ページで編集操作を表現しなければならかったため使い勝手が複雑化するが、XML-P'z 文書を用いれば、複数ページに分割した表現が可能であるため、使いやすいものにできる。またページ間でのデータのやり取りができることから、Web サイト内にとどまらず、Web サイトを越えたコンテンツ連携が可能になる。これにより、所望のコンテンツをサイトを越えてシームレスに収集したり、ピースを組み合わせて、新しい文書を作成するなどの作業を直感的に行うことができる(図 3)。

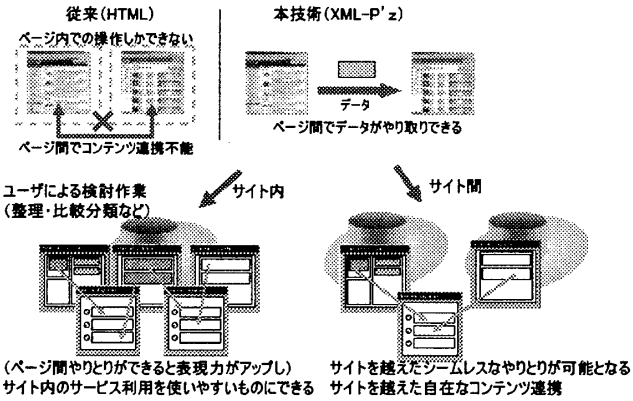

図 3: HTML と XML-P'z の操作性比較

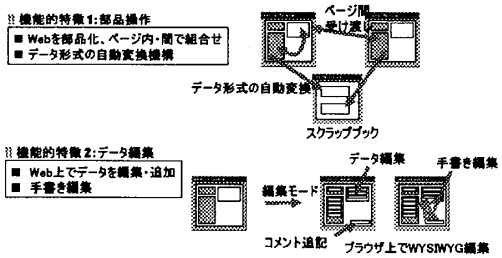

図 4: 基本編集機能

またコンテクストメニューから編集モードを起動すると、(スタイルを通じて)表示されているピースのビューのうち、内部の XML ノードに対応する記述箇所が編集可能な状態となり、WYSIWYG でコンテンツを編集できる。これにより、内容を更新したりコメントを追記するなどが可能であり、単にあちこちからコンテンツを集めてくるだけでなく、新しいコンテンツ作成が可能となる。図 4 は以上の GUI 操作をまとめたものである。

[クラス定義による追加機能群]

ピースに対して複数のスタイルが与えられている場合、コンテクストメニューを通じてピース毎に表示を切り替えることができる。これにより、たとえば商品同士をデザイン中心・スペックで比較するなど、様々な表示方法を通じた多面的な検討が可能となる。内部の XML データとスタイルの 2 重構造となっていることから、データとしてはすべての項目を維持するも、各スタイルでは、比較検討において必要な項目だけを表示すればよい。

ピースに対してメソッドが与えられている場合、コンテクストメニューを通じてメソッドを呼び出すことができる。たとえば商品カタログに購入メソッドが用意されているなど、通常アプリケーションに添った形で提供されたカスタムメニューである。また設定されていれば、通常 HTML のフォームのようにスタイル上の GUI ボタンなどからメソッドを呼び出すこともできる。またメソッドは他ピースのメソッドを呼び出せるので、そのように設計されていれば、商品にオプションとして挿入したサブ商品群についても、まとめて購入申請するといったことも可能である。

ピースに対してコンストラクタが与えられている場合、任意パネル上のコンテクストメニューから新規作成コマンドを実行すると、ピースを新規作成することができる。自動的に編集モードとなり、構造に沿ったコンテンツ記述が可能である。これにより、コンテンツを集めてくるだけでなく、新たにコンテンツを作り出すことができる。

図 5 は以上の GUI 操作をまとめたものである。

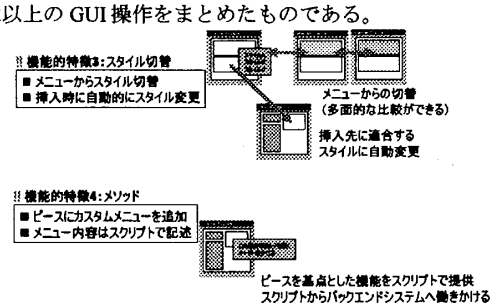

図 5: クラス定義による追加機能群

4. 安全なコンテンツ利用の設計

Visual eXcart は、前節のとおり、Web サイトを越えてコンテンツをやりとり可能にする分散コンテンツのための基盤技術である。システム間の連携性を向上させることはメリットである一方、信頼できないシステム間とのやりとりが行われる可能性を持つため、各コンテンツやアプリケーションの動作性や安全性の保証において問題がある。

そこで、Visual eXcart ではこの問題に対処するため、利用制御とジャーナルという 2 つの機能を具備している。利用制御とは分散コンテンツの利用環境にあって、提供されたコンテンツやアプリケーションを想定目的に沿って正しく使うように制御する機能であり、利用制御を用いることで、XML-P'z 言語上でコンテンツの利用方法を事前定義することが可能である。ジャーナルとは操作履歴を行う機能である。これにより事前定義された利用制御の網をくぐって発生した問題に対して、事後収集のために過去を追跡確認することができる。ジャーナルは、各ピース内およびサーバに格納される。これにより、たとえばピースがどの URL から取得したものなのか判定でき、ピースが誰にいつコンテンツ編集されたのかを調べることができる。ジャーナルについては以上の説明に留める。

4.1 利用制御の基本方式

利用制御は次のような基本方式を採用している(図 6)。すなわち、提供する各リソースに対して使い方の設定(これを利用制約と呼ぶ)を付与する形で配布し、XML-P'z パーサである Visual eXcart は、リソースに付与された各利用制約をチェックして、制約に合致した編集機能のみをユーザに許可する。

利用制御には大きく、ピース利用制約とパネル利用制約の 2 種類がある。ピース利用制約はピースの利用方法を規定する制約である。これによりたとえば、ピースの配布流通範囲を制限したり、コンテンツ書換えや印刷を禁止するなど機能を制限するなど、ピースのコンテンツ利用を保護する。挿入適用先の文書のケースごとに文脈つきで設定できる。パネル利用制約はパネルの利用方法を規定する制約である。これによりたとえば、文書内の各パネルごとにどのようなピースを受理するのか、また受理した場合、そのパネル内ではどのようなピース利用を許可するのか、などの設定ができる。これにより、構造文書(アプリケーション)への不正入力を防止するとともに、各パネルへの役割分担を与えるなどの構造設計が可能となる。

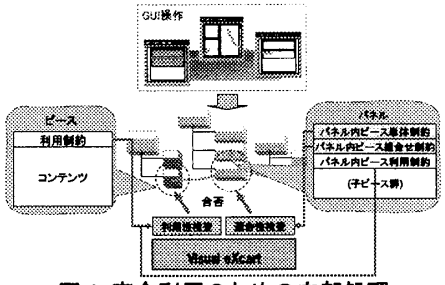

図 6: 安全利用のための内部処理

利用制御が提供する制約の種類は以下の構成である。

- ピース利用制約
  - 状態ケース
  - ロケーションケース
  - ピース利用制約条件
- パネル利用制約
  - パネル内ピース単体制約
  - パネル内ピース組合せ制約
  - パネル内ピース利用制約

4.2 ピース利用制約の定義

ピース利用制約を定義するには、クラス定義内において、状態ケース指定子・ロケーションケース指定子・ピース利用制約子を用いる。以下は定義例である。

```
<pz:class pz:name="Catalog">
 .. (snip) ..
 <pz:case pz:stateif="test(//@secret['true'])">
  <pz:location>
   <pz:allowstyle  pz:list="style:name(s)"/>
   <pz:allowaccess pz:default="none"/>
  </pz:location>
 <pz:case pz:stateif="">
  <pz:location pz:rootif="url(http://a.com)">
   <pz:allowstyle  pz:list="style:name(a1):style:name(a2)"/>
   <pz:allowmethod pz:list="method:name(buy)"/>
   <pz:allowaccess pz:default="read"/>
  </pz:location>
  <pz:location pz:rootif="url(http://b.com)">
   <pz:allowstyle  pz:list="style:name(b1):style:name(b2)"/>
   <pz:allowmethod pz:list=""/>
   <pz:allowaccess pz:default="read"/>
  </pz:location>
 </pz:case>
</pz:class>
```

ピース利用制約はクラスメンバーに記述し、状態ケース指定子(pz:case)・ロケーションケース指定子(pz:location)・ピース利用制約条件子(pz:allow***)の 3 重構造を取る。以下概要を説明する。

状態ケース指定子は、ピース自身の状態、すなわちピースのコンテンツ値によってケース分けを行う(これを「状態ケース」と呼ぶ)。上記例では、コンテンツ内の属性 secret の値が true かどうかでケースわけを行っている。2 つ目の状態ケースでは条件が付与

されていないが、これをデフォルト状態ケースと呼び、すべての状態にヒットする。

ロケーションケース指定子は、ピース外の利用環境、すなわちピースの収納先パネルや文書によってケース分けすることができる(これを「ロケーションケース」と呼ぶ)。上記例では、デフォルト状態ケース内で、収納先文書の Web サイトによって 2 つのロケーションケースを定義している。ピースはロケーションケースに指定されていない文書では生存できない。これにより、流通範囲を限定できる。前記サンプルでの 2 つ目のロケーションケースには条件が付与されていないが、これをデフォルトロケーションケースと呼ぶ。デフォルトロケーションケースはすべてのロケーションにヒットする。

ピース利用制約条件子は、指定ケース内で利用許可するピースの機能群を定義する。secret 属性が true の状態ケース下のデフォルトロケーションケースでは、スタイル s のみ許可するとともに、コンテンツへのアクセスは読み書き不可となっている。この場合、墨塗りされた各コンテンツ値がスタイル s を通じて表示される。

4.3 パネル利用制約の定義

パネル利用定義には、パネル内ピース単体制約、パネル内ピース組合せ制約、パネル内ピース利用制約があり、パネル指定子の属性または子エレメントである、パネル単体制約子(@pz:for)、パネル内ピース組合せ制約子(@pz:constraint)パネル内ピース利用制約子(pz:usage)で定義する。それぞれ独立した制約である。

パネル単体制約は、パネルが受理する各ピース個別に対して満たすべき性質を規定する制約である。パネル単体制約を付与すると、たとえば、この制約条件を逸脱するようなピースのドラッグドロップ挿入を拒否したり、格納中ピースが条件逸脱するようなコンテンツ編集をキャンセルする。

パネル内ピース組合せ制約は、パネルが受理するピースの集合に対して課す制約である。たとえば、ピースの順序入れ替えを禁止したり、最大収納数を設定することなどが可能である。

パネル内ピース利用制約は、ピースが自パネル内に収納されているときに限り、パネルが受理している各ピースに対して、追加的なピース利用制約を課すものである。

以下は定義例である。

```
<pz:panel pz:name="goods"
 pz:for="class:uri(classes.pzx#pzclass(Catalog))
and instance:url(http://a.jp)"
 pz:constraint="panel:maxoccur(5):panel:rotate(true)">
 <pz:usage>
 <pz:allowaccess pz:default="read"/>
 <pz:allowstyle  pz:available="style:name(x1):style:name(x2)"/>
 </pz:usage>
</pz:panel>
```

パネル goods は、パネル内ピース単体制約により、a.jp サイトが配布元である Catalog クラスのピースのみを受理するよう制約される。またパネル内ピース組合せ制約により、最大収納数は 5 個であり、それを越える挿入があった場合は最後方のピースと入れ替える。またパネル内では、カタログの編集は禁止するとともに(自社スタイルである)x1 および x2 での表示のみ許可する。なお、ピースの配布元 URL はジャーナル機能により記録されている。

4.4 利用性検査と適合性検査

利用性検査は、ピースに対して利用できる機能の可否を判定する処理である。利用性検査は、ピースに付与されたピース利用制約のうち現在ケースにマッチするピース利用制約と、ピース収納先パネルに付与されたパネル内ピース利用制約を合成した結果を用いる。利用性検査は、ピースの機能を実行する直前のタイミングで実行される。この検査結果によって、利用できる機能やメニュー構成が変化する。

適合性検査とは、ピースがパネル内に存在可能かどうかを判定する処理である。適合性検査は、ロケーションケース、パネル内ピース単体制約、パネル内ピース組合せ制約、の 3 つの制約に影響を与える操作が実行された後に、各制約条件について実行される。試行操作と適合性検査で用いる制約条件は、次のとおりである。この検査に不合格した場合は、操作がキャンセルされ、実行前に戻る。

- ケース1: ピース兄弟関係の変化
  試行操作: 新規作成・削除・パネル内移動・コピー
  適用制約: パネル内ピース組合せ制約
- ケース2: ピース内部状態(コンテンツ値)の変化
  試行操作: コンテンツ編集・XML-Pzメソッド
  適用制約: パネル内ピース単体制約
- ケース3: ピース外部環境(祖先パネル)の変化
  試行操作: パネル間移動・コピー
  適用制約: ロケーションケース・パネル内ピース単体制約・パネル内ピース組合せ制約

以上の検査処理により、想定外の利用方法を未然に防止することができる。

5. 適用例: 社内 OA 購入申請システム

図 7: 社内 OA 購入手配システムへの適用

図 7 は、社内での OA 購入手配システムに Visual eXcart を適用したものである。購入物品群・手配先業者・使用予算枠など、申請に必要な各項目欄について、製品カタログデータベース、資材業者データベース、予算管理データベースなどから選択的に引用する。引用に伴うユーザの操作はドラッグ＆ドロップのみだが、先ほど述べた 3 つの安全性を提供している。1 つはデータ整合性であり、購入物品の種別が予算や手配先に照らして適切かどうか、購入物品の合計金額が予算を超過していないかなどをそのつど検査し、無効な場合は拒絶する。これにより、インタラクティブに試行錯誤しながら、間違いのない申請書を直感的かつ効率よく作成できる。2 つ目は真正性の確保である。根拠データとして引用した各データについて、その引用元を特定して捏造を防止する。3 つ目は漏洩防止である。文書の多目的利用として、作成した申請書に含まれる各データを、たとえば業者への見積もり依頼書など他の書類作成において再利用できるようになっているが、書類の用途に応じて、社外秘情報など開示すべきでないデータ項目は自動的に除去される。

6. おわりに

Visual eXcart は「集めて」「使う」「安全に」というコンセプトを元に編集作業を支援するシステムである。すなわち、Web システムに散在するデータを目的に沿って収集・比較分類、整理結果に基づいてデータを再利用し、さまざまな用途の文書を効率よく 2 次作成、さらに生後製チェック・エビデンス生成、朗詠防止などガイドラインに沿って安全に利用を誘導できる。これらの特徴から、システム内外の情報の安全かつ簡便な活用を実現する。

[*1] 松岡正剛 : 知の編集工学 朝日新聞社, ISBN:4022613254, 2001/02