Vol.2014-DBS-159 No.16
Vol.2014-IFAT-115 No.16
2014/8/2

非技術者によるXMLサブセットデータ入力を容易にするファイル形式”XYML”

井戸 伸彦[1,a)]

概要：非技術者にも XML サブセットデータの入力が容易に行えることを目的とした，ファイル形式”XYML(Xml in YaML format)”を提案する．このようなファイル形式は既にいくつもの提案があるが，本稿で提案する XYML はデータ直列化形式である YAML ファイルとして読み書きが出来ることに大きな特徴がある．XML のツリー構造データの簡明さと YAML の可読性の良さとを活かすことにより，目次風の分かりやすいファイル形式を XYML は実現している．４０名の学生を被験者として行った可読性の評価実験では，データに関する問いへの平均解答時間が XYML は XML よりも２０％程度短く，有意水準５％でこの差異があるとの検定結果を得た．また，プログラミング言語 Ruby での XYML 形式ファイルを扱う API を定義してパッケージを実装した．このパッケージでは，配列とハッシュとの交互の入れ子としてツリー構造のオンラインデータを実現し，これの直列化形式が XYML となるような実装としている．

1. はじめに

XML が広く普及すると同時に，次のような問題点も指摘され，これらの問題点に対応する提案も行われている [1]．**（性能に関する問題点）** XML は冗長で複雑であり，XML 文書の処理を行う端末等での負担が大きい [2]．近年のモバイル端末の急速な普及に伴い，消費電力等を抑える必要のある環境での利用を念頭においた提案も相次いでいる [3][4]．**（可読性に関する問題点）** マークアップ言語である XML は，「テキストファイルとして扱える」という意味での可読性は備えているが，「人が読み書きしやすい」という意味では難がある．これに対しては，XML と意味的には同じとなる省略構文が提案されている．文献 [5] では”アダプター”と記されているこのような省略構文には，PXY，SOX，SXML などがある．

一方，上記のような問題点を踏まえて優劣が比較されるものに，YAML[6]，JSON などのデータ直列化形式がある．YAML の仕様書 1.4 項（[6]YAML1.2）に記されているように，XML は構造化文書として設計された SGML を引き継いでおり，YAML とは直接対比は出来ない．しかしながら，XML と YAML とはいくつかの分野では競合関係にあり，その競合関係の中で YAML は”human friendly”であることを唱っている．なお，JSON は”フロースタイル”として YAML の規格に取り入れられている．以下では，YAML と記す際はその”ブロックスタイル”を指す．

一般に，システムごとに専用の入力ツールや閲覧ツールを必要としない点で，テキストベースの可読性の高いデータ形式は大きな魅力があり，XML と YAML はこの点で今後とも大きな役割を果たすと期待される．しかしながら，技術者ではない操作者がテキストエディタで入力を行う場面を考えると，2.1 項で言及するように XML と YAML の両者ともに問題点は残る．

本稿では，非技術者がテキストエディタで入力する際に都合が良いことを目的としたファイル形式についての提案を行う．この目的に関して XML や YAML とは全く別のファイル形式を新たに提案することは，屋上屋を架すことになるのみで，現実的な意義は乏しい．本稿で提案するファイル形式の特徴は，”YAML として読み書き出来，XML として利用出来る”ことである．YAML として読み書き出来ることが直接可読性に結びつく訳では無いが，入力されたデータを扱うシステムの開発者の立場からは，XML ／ YAML の既存技術や認知度を活かすことが出来るという利点がある．提案するファイル形式を XYML（Xml in YaML format）と記すこととする．

XYML は，現在筆者が開発を進めている手書き文字入力により漢字検定相当の試験を実施する web システム [14][15] での利用を目的として考案した．このシステムでは，学校教員が試験問題を作成してアップロードする際のファイル形式として XYML を利用する．この試験問題データは，正答・別解や配点，誤答に対するフィードバック等のデータ

[1] 岐阜経済大学経営学部，大垣市
Gifu Keizai University, Ogaki-shi, 503-8550 Japan
[a)] ido@gifu-keizai.ac.jp

を含んでおり，構造化文書ではなく，木構造データである．

以下，2 項では，XML/YAML の仕様を比較しながら，XYML の仕様の概略をその考え方と共に記す．3 項では，プログラミング言語 Ruby で行った XYML パッケージの実装について記す．4 項では，XYML の可読性について XML との比較において行った評価実験について記す．

2. XYML の提案

2.1 XML/YAML との比較

表 1 に，XML/YAML の比較と，提案した XML がいずれの仕様を引き継ぐかを示す．端的に言えば，XML では開始・終了タグによるノードの記述が読みづらいこと，YAML ではツリー構造を形成するデータ形式 (配列，ハッシュ等) が非技術者には馴染みにくいことが，主要な問題点となる．提案する XYML では，これらの問題点を避けた形で，双方の仕様を引き継いでいる．

図 1 に，XYML の位置づけを示す．表 1 に示した XML/YAML との比較をまとめると，図 1 のように「XYML は，データ構造は XML，表記は YAML」ということになる．さらにデータ構造は XML であるので，XML のサブセットとなる表記への変換を可能としている．

2.2 XML の構成単位との対応付け

XYML は，XML の構造を YAML の表記に対応付けしたものである．その対応付の大きな特徴は，XML の構成単位である”要素”,”属性”,”テキスト”を，YAML の構成単位である”ハッシュ”,”配列”,”スカラー”の組み合わせに対応付けしていることである．すなわち，構成単位の組み合わせだけを用いた対応付けであり，予約語とするキーワードを導入して一方が他方を記述するタイプの変換ではない（例えば，YAML[6] 仕様書 1.4 項で言及されている，YAML データを XML での記述に変換する YAXML はキーワードを導入した変換である）．

図 2 に，XML から XYML（YAML）へのマッピングを示す．図 2 に示す対応は，XML の構成単位を YAML の構成単位に対応付けるものであるが，XYML から XML への対応も次のように一意に決まる．

(1) 原則，すべての行は”-”（ハイフン）で始まる．
(2) ”:”で終わるハッシュのキー名を示す行は，XML の要素名に対応する．その要素の子要素や属性は，次行以降に該ハッシュのバリューである配列に記される．
(3) ”:”によりハッシュのキー名とバリュー値に分けられた行は，XML の属性名と属性値に対応する．

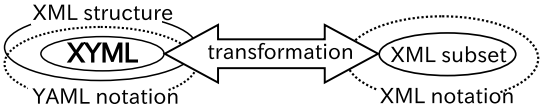


図 1 XYML の位置づけ

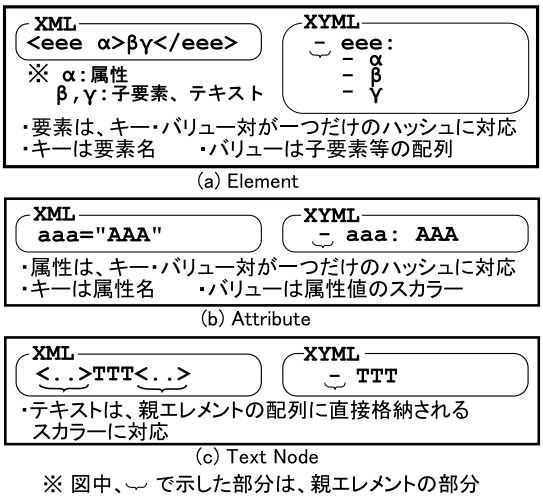


図 2 XML から XYML（YAML）へのマッピング

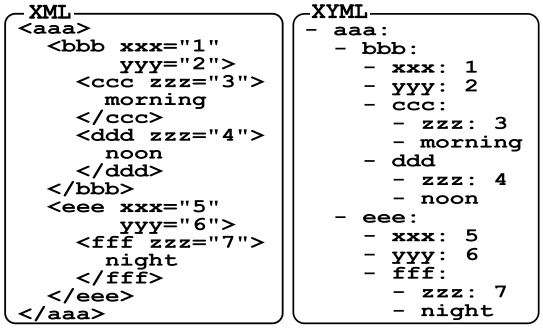


図 3 マッピングの例 -1-

(4) ”:”（コロン）を含まない行は，テキストに対応する．

XYML データの，YAML 表記としてのルートは，長さが 1 の配列とする．これにより，XYML のルート要素の行も先頭は”-”（ハイフン）となり，原則すべての行がハイフンで始まることになる．この単純さは非技術者にとって分かりやすく，XYML を本の目次や箇条書きと捉えた際の表記としても自然である．

2.3 対応する XML/XYML の例

図 3 に，属性とテキストとが混在した XML と，これにほぼ等価な XYML との例を示す．図中の XML では，可読性向上のためにインデントを挿入し，2 つ以上の属性がある場合はその間で改行している．両者が完全に等価でないのは，インデントの空白文字が XML のテキストに含まれているためである．XYML では，そのような空白文字を扱う必要が無い．

図 4 に，Tomcat7[12] で使用される”server.xml”ファイルからの抜粋である XML の記述例と，これに対応する XYML の記述例を示す．パラメータの指定に属性のみを用いていている図の XML では，最初の 3 行以外はコメントである．図中の XYML では，これと同じ意味となるコメントを付加している．図の例で XML が無駄の多いコメントを行っているのは，XML では属性に係るコメントが

表 1　XML/YAML の比較と XYML

	XML	YAML	XYML
ツリー構造	単純なツリー構造 ○テキストなどの葉ノードを除き、すべてクラス名相当の名前を有するノード.	配列とハッシュとの混合によるツリー構造 ×非技術者には、配列・ハッシュの概念は自明でない. → 行頭の"-"(ハイフン)の要不要がわかりにくい.	XML
ノードの記述方法	記述的マークアップ言語(文献[7]) ×開始タグと終了タグを用いる方法が、可読性に対する不満の原因となっている.	記述的マークアップ言語、一部表示的[7] ○改行やインデントによる表示的な意味合いも持ち、これにより可読性が向上している.	YAML
スキーマ	データモデル(文法)が異なる複数の方法がある[9]. ・DTD(local tree) ・XML Schema(single-type tree)・RELAX NG(regular tree)	Kwality[8]等を除くと、議論・実用とも少ない.	XML (RELAX NG,∗2)
インライン要素	○自然に記述出来る.	×一般的な記述は不可(Movable Type[11]等のように、テキスト中にXMLのインライン要素を記すことは可能).	一般的な記述も可とする(∗1).
(∗1)XML-XYML変換を可能とするために、可読性には劣るが一般的なインライン要素の記法を仕様に含む(詳細略).また、YAMLと同様に、テキスト中にXMLのインライン要素を記すことは可能. (∗2)実装においては、XYMLからXMLに変換し,RELAX NGで妥当性を検証する.			

XML
```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```

XYML
```
- Connector
  # uncommnent when using the shared thread pool
  # - executor: tomcatThreadPool
  - port: 8080
  - protocol: HTTP/1.1
  - connectionTimeout: 20000
  - redirectPort: 8443
```

図 4　マッピングの例 -2- (Tomcat7 の server.xml より)

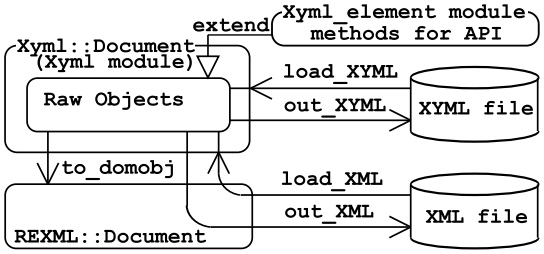


図 5　XYML パッケージの実装の概略

付加出来ないためである．これに対して，XYML では属性に自然にコメントが付加出来る．

2.4　サブセット化と制限，拡張

図 1 に示したように，XYML は YAML のサブセットであり，XYML と等価な XML は，XML 全体仕様のサブセットである．すなわち，YAML でのハッシュの複合キーとそのキーインディケーター”?”など，YAML の一部の仕様 XYML の仕様に含まれない．また，XML の実体参照，処理命令，CDATA に対応する仕様は，XYML の仕様に含まれない．なお，名前空間は記述可能であるが，3 項に示す実装では，現状関連機能は盛り込んでいない．

また，XYML では要素と属性との表記は似ており，「属性が先で子要素は後に記述するという制約は無くす」などの様々な拡張を行える可能性を秘めているが，本稿ではこれらについては触れない．

3.　Ruby での実装

前節で提案した XYML ファイル形式を，プログラミング言語 Ruby で扱うためのパッケージの実装を行った（[16]）．

3.1　パッケージの概略

図 5 に，XYML パッケージの実装の概略を示す．パッケージは，`Xyml::Document` クラスを含む `Xyml` モジュールと，XYML のオンラインデータへの API を定義する `Xyml_element` モジュールとからなる．`Xyml::Document` クラスは，XYML ファイルに加えて XML ファイルとの入出力機能を備える．これらの機能は，Ruby の標準 XML パッケージである `REXML` パッケージや `YAML` パッケージを利用して実現されている．

次節以下に，図 5 に示す実装の特徴について記す．

3.2　配列とハッシュによるツリー構造

今回行った実装では，直列化した際に XYML ファイルとなるオブジェクト (図 5 中の”Raw Objects”) を，オンラインのツリー構造データの形式としている．実際，図 5 中の `load_XYML` では，Ruby の `YAML` パッケージを用いてファイルのロードを行っている．この結果，オンラインでのツリー構造データは，2.2 項に記した対応に基づいて配列とハッシュにより保持される．Ruby では，配列（`[a,b,c]` の形式），ハッシュ（`{a:A,b:B,c:C}` の形式，但し，`a,b,c` はシンボル）とも，ネイティブデータとして扱えるため，極めてシンプルなツリー構造データの表現となる．この結果，図 3 に示したファイルを出力するためのオンラインデータの初期化は，図 6 左側に示すプログラムで行うことが出来る．このプログラムでは，データ構造を確認する `pp` メソッドにより作成したツリー構造データを出力しているが，図 6 右側に示すように，その出力結果でも初期設定と

```
Program(Ruby)
require 'xyml'
tree=Xyml::Document.new(
  {a:[{b:[{xxx:1},
          {yyy:2},
          {c:[{zzz:3},
              "morning"]},
          {d:[{zzz:4},
              "noon"]}]},
      {e:[{xxx:5},
          {yyy:6},
          {f:[{zzz:7},
              "night"]}]}]})
tree.out_XYML(
  File.open('a.xyml','w'))
pp tree
```

```
Results
[{:a=>
  [{:b=>
    [{:xxx=>"1"},
     {:yyy=>"2"},
     {:c=>[{:zzz=>"3"},
           "morning"],
      :parent=>{...}},
     {:d=>[{:zzz=>"4"},
           "noon"],
      :parent=>{...}}],
    :parent=>{...}},
   {:e=>
      (中略)
    :parent=>{...}}],
  :parent=>:iamroot}]
```

図 6　ツリー構造データの初期化

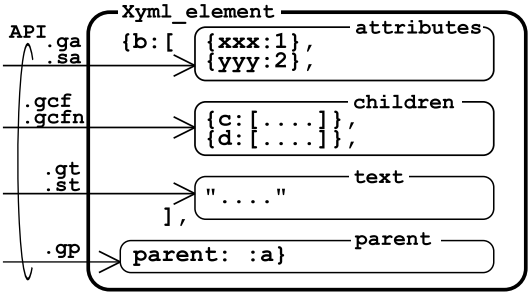


図 7　API 向けの XYML 要素のモデル

同様なシンプルな表示が得られる．

3.3　要素を中心とした単純なモデルに基づく API

3.2 節に示した配列とハッシュによるツリー構造データは，普通の配列・ハッシュとしてアクセスすることが可能であるが，それは煩雑である．このツリー構造データへアクセスする API を，図 5 中の”xyml_element”モジュールとして実装した．これは，XML の DOM API に相当するものである．しかしながら，図 5 に示したように，XYML ファイルのデータは，DOM API を提供する REXML::Document インスタンスに変換することが出来るので，DOM API に似た API を作成しても意義は小さい．また，REXML パッケージのように，従来の DOM API は，Element,Parent,Child といったクラス間の継承関係に基づいたものになっている．これらは，オブジェクト指向の観点からは合理的とも考えられるが，単純なデータ構造であるツリーに対する整理方法としては煩雑とも考えられる．

今回の実装では，図 7 に示すように要素だけをモデル化して，これへのインタフェースを提供するシンプルな方法を採った．図 7 中の，”parent”をキーとするハッシュのバリューは，親要素を得るためのデータであり，Xyml::Document インスタンス生成時に付加される．図 6 中の実行結果にも，これが表示されている．XYML ファイルを保存する (out_XYML) 際には，このキー・バリュー対を取り除いている．

```
tree_data.root.gcfn(:b)
```

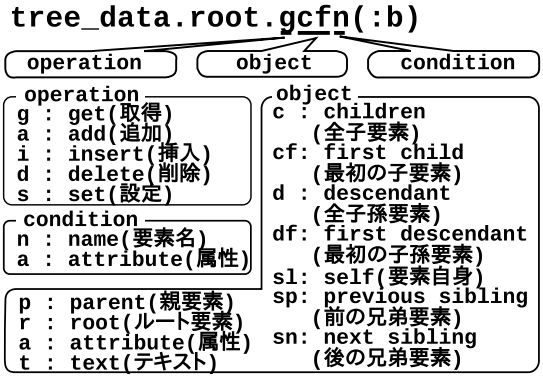


図 8　API の命名規則

図 8 に，API の命名規則を示す．図に示すとおり，メソッド名は，操作 (operation)・対象 (object)・条件 (condition) を指定する文字列を順に並べた規則的なものである．Ruby でのメソッドチェイン（メソッドを連続的に呼び出す方法）を活かすために，このような短いメソッド名とした．例えば，図 6 のプログラムで，要素”d”の属性”zzz”の値である”4”を得て打ち出すには，次のように書けば良い．

```
p xyml_tree.root.gcfn(:b).gcfn(:d).ga(:zzz)
```

API については，XPath 相当の導入や，2.4 項（５）に記した拡張などについて，今後検討していきたい．

4.　評価

提案したファイル形式 XYML について，XML との比較における可読性に焦点を当てて評価実験を行った．

4.1　可読性の評価方法

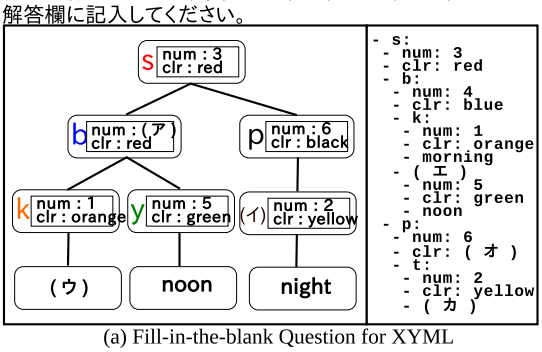
可読性（readability，文章の読みやすさ）の客観的評価としては，読むことに要する時間を計測する方法が分かりやすい [13]．また，XYML と XML とは，上位概念としてツリー構造データを持つことが共通している．これらを踏まえ，ツリー構造として図示されたデータを XYML と XML とのそれぞれに対応付ける問いに対する解答時間により評価を行うこととした．図 9 に，評価で用いた XYML 向けの問いを示す．図中，(a) が穴埋め問題，(b) が記述問題となっている．XML 向けの問いは，全く同じデータとその図示を用い，穴埋め空欄も同じ位置としている．出題・解答は web ページで行う．

4.2　実験の実施方法

筆者が勤務する文系大学の学生に任意の協力を求め，４０名の被験者を得て実験を行った．被験者は，図 10 に示すＡとＢとの二つの出題パターンのいずれかについて解答を行う．パターンＡは，図 9 に記した XYML 向けの問い（図 10 での (a.1)(aw.1)）に先に答え，その後に構造と値が異なる別のデータについて XML 向けの類題（(a.2)(aw.2)）に答える．パターンＢは XML が先（(b.1)(bw.1)）で，XYML

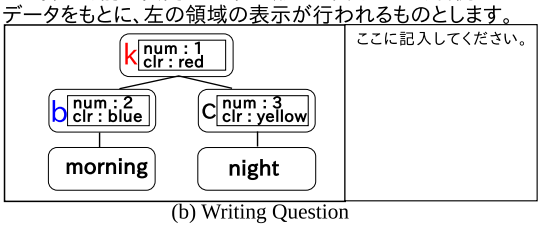
Vol.2014-DBS-159 No.16
Vol.2014-IFAT-115 No.16
2014/8/2

質問1.1　次の図中、右の領域のデータをもとに、左の領域の表示が行われています。(　ア　)～(　カ　)には何が入るか、解答欄に記入してください。



(a) Fill-in-the-blank Question for XYML

質問1.2　次の図中の、右の領域のデータを記入してください。但し、前の質問1.1と同じ形式・規則により、右側のデータをもとに、左の領域の表示が行われるものとします。



(b) Writing Question

図 9　評価で用いた問い（XYML 向け）

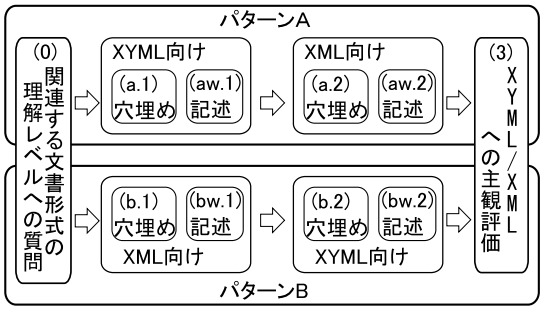


図 10　評価実験での２つの出題パターン

表 2　関連する文書形式の理解レベル

用語	パターン	聞いたことが無い	聞いたことはある	内容を知っている	勉強したことがある
HTML	A	0	6	3	11
	B	0	9	4	7
XML	A	3	12	3	2
	B	9	6	1	4
YAML	A	17	3	0	0
	B	18	1	1	0

が後（(b.2)(bw.2)）の順で答える．(a.1) と同じ構造と値を持つデータに対して XML を用いた解答を求めるものが (b.1) であり，正解は全く同じものになる．(a.2) と (b.2) とも同じ関係にある．記述問題の (aw.1) と (bw.1) とは，図 9(b) に記した同じ構造と値を持つデータに対してそれぞれ XYML と XML との形式で記述することを求めるものである．A ／ B 両パターンとも，最初に図 10 中（０）関連知識に関する質問と，最後に（３）XYML/XML への主観評価とを行っている．

図 10 中，(a.1) と (b.1) とは対応の無いデータとして統計的検定の対象となる．類題である (a.1) を実施した後に (a.2) を実施するので，(a.1) と (a.2) とは統計的検定の対象とはならない．一方，(a.2) と (b.2) とは，XYML と XML とが入れ替わっているが，ともに類題を実施した後であるという意味で同等な条件であり，統計的検定の対象となる．

図 10 での (a.1) もしくは (b.1) の採点結果が，図 9(a) に対応する 6 問中 4 問以上の正解したものだけを統計の対象としたが，その合計が上記の４０名である．(a.1) の平均正解数は 5.6，(b.1) は 5.65 で両者に大きな違いは無い．表 2 に，図 10 中（０）の，関連する文書形式の理解レベルへの質問の結果を示す．表に示すように，これについても A と B とのパターンで被験者に大きな違いは見られない．

実験はすべて大学内の情報機器を備えた教室で筆者立ち合いのもとで行い，個人のアカウントとパスワードにより被験者個人を特定している．

4.3　解答時間による客観評価結果

図 11 に，実験での穴埋め問題の解答時間の分布を示す．図中の (a.1)(a.2)(b.1)(b.2) は，それぞれ図 10 内の表記に対応する．パターンＡとＢとの各々について２０名の被験者が 6 問の設問に答えており，合計１２０問の回答時間の分布となっている．解答時間の平均は，第 1 問めの XYML の (a.1) が 23.23 秒，XML の (b.1) が 30.34 秒であり，同じ設問に XYML の方が短時間で解答出来るという意味で，可読性が高いと言える．両者の平均に差が無いとする帰無仮説を welch の t 検定（対応の無いデータ）を適用すると p 値は 0.033 となる．すなわち，有意水準 5 %で両者の解答時間の平均値は異なると言える．

第 2 問めの XYML の (b.2) の 14.04 秒と XML の (a.2) の 21.25 秒との平均の解答時間の比較においても，同じ帰無仮説に対して p 値は 0.0039 となる．すなわち，1 問めで「XYML の A グループ」が「XML の B グループ」に勝り，２問めで「XYML の B グループ」が「XML の A グループ」に勝るという，強い結果になっている．なお，実験を途中で離脱した被験者もいるため，(a.2)(b.2) は (a.1)(b.1) よりも，標本サイズが小さくなっている．

図 12 に，穴埋め問題の平均解答時間の詳細を示す．図 9 に示したように，穴埋め問題には，要素名を答えるもの，テキストを答えるもの，属性値を答えるものがそれぞれ 2 問ずつあり，これらで分けた場合の平均値が図 12 に示されている．いずれにおいても XYML が XML よりも短い解答時間となっており，特別な部分で差が生じている訳でなく，記法全体として XYML が優位であることが窺える．

図 13 に，穴埋め問題での解答欄へのアクセス回数の分布を示す．穴埋め問題は 6 問で構成されるので，後戻り無く順に解答欄に入力を行うとすると，アクセス回数は 6 回となる．誤りに気付いて訂正したり，自信が無い解答を再度確認したりする行為があると，アクセス回数は増える

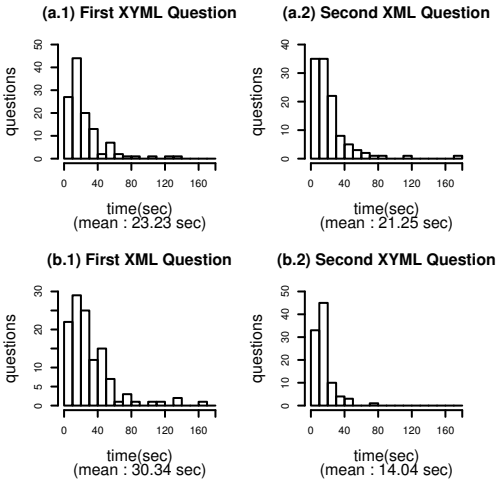


図 11　穴埋め問題の解答時間の分布

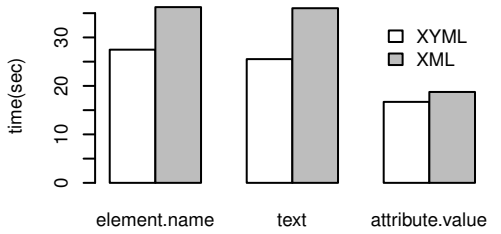


図 12　穴埋め問題の平均解答時間の詳細

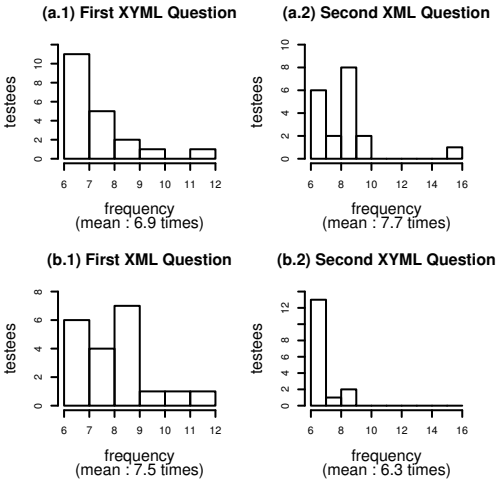


図 13　穴埋め問題での解答欄へのアクセス回数の分布

と考えられる．図に示すように，XYML の (a.1)6.9 回と (b.2)6.3 回とが，対応する XML の (b.1)7.5 回と (a.2)7.7 回よりもアクセス回数の平均値がそれぞれ少なくなっている．平均に差が無いとする帰無仮説の welch の t 検定では，それぞれ p 値が 0.19，0.0094 となり，有意な結論は導けないが，XYML の優位性を窺わせるデータとなっている．

記述問題については，図 10 の (aw.1) の平均解答時間は (bw.1) のものよりも短いという結果であったが，実験前には想起していなかったコピー＆ペーストを利用するか否かかが解答時間に大きな影響を与えている様子であったため，これを XYML の優位性の根拠とすることは出来なかった．

4.4　主観評価結果

図 10 中（３）の XYML/XML への主観評価では，いずれの書き方が分かり良いと感じたかについて二者択一での回答を求めた．結果は，XYML を分かり良いとする被験者が 22 名，XML が 18 名となり，客観評価結果から期待されるような大きな差はつかなかった．

5.　おわりに

本報告では，提案した XYML ファイル形式の可読性を実験での客観評価により示すことが出来たが，普及の可能性を探るには実用での評価が必要となる．前述（1 項）の手書き文字入力により漢字検定相当の試験を実施する web システムの実働に伴い，実用レベルでの XYML への評価を行う予定である．

参考文献

[1] P. Shadiya and P.A. Haleem, “Energy efficient data formatting scheme: A review and analysis on xml alternatives,” IJAIS, vol.1, no.1, pp.10–13, 2012.
[2] H. Liefke and D. Suciu, “Xmill: an efficient compressor for xml data,” ACM SIGMOD Record, vol.29, pp.153–164, June 2000.
[3] M. Girardot and N. Sundaresan, “Millau: an encoding format for efficient representation and exchange of xml over the web,” Computer Networks, vol.33, no.1, pp.747–765, 2000.
[4] P.A. Haleem and M. Sebastian, “An energy-conserving approach for data formatting and trusted document exchange in resource-constrained networks,” Knowledge and information systems, vol.32, no.3, pp.559–587, Sept. 2012.
[5] N. Ershov, “Tabula language for description of structured data,” Moscow University Computational Mathematics and Cybernetics, vol.33, no.4, pp.214–218, 2009.
[6] http://www.yaml.org/
[7] J.H. Coombs, A.H. Renear, and S.J. DeRose, “Markup systems and the future of scholarly text processing,” Communications of the ACM, vol.30, no.11, pp.933–947, 1987.
[8] http://rubygems.org/gems/kwalify
[9] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, “Taxonomy of xml schema languages using formal language theory,” ACM Transactions on Internet Technology (TOIT), vol.5, no.4, pp.660–704, 2005.
[10] E. Van derVlist, Relax Ng, O’Reilly, 2011.
[11] http://www.movabletype.org/
[12] http://tomcat.apache.org/
[13] 山口徳郎，他，“利用者とディスプレイの位置関係を考慮したパースペクティブ表示，” 信学論（D），vol.J91-D，no.12，第 J91-D 巻，pp.2746–2754，Dec. 2008.
[14] 井戸伸彦，“特徴表示を伴う手書き文字入力とこれを利用した自動採点システム，” 信学技報，ET2012-79，no.79，pp.57–62，Jan. 2013.
[15] 井戸伸彦，“多軸順序距離を用いた手書き漢字の画の対応付け，” 信学技報，PRMU2014-16，no.41，pp.85–90，May 2014.
[16] https://rubygems.org/gems/ixyml
https://github.com/nobuhiko-ido/Ixyml