Parallel writing in East Asian languages and its representation in metadata in light of the DCMI Abstract Model

Akira Miyazawa
NII, Japan
miyazawa@nii.ac.jp

Abstract

This paper discusses the parallel writing tradition in East Asian languages and its representation in metadata. Parallel writing systems in these languages do not use the same scripts, but they all share a common scheme and have a well-established tradition in bibliographic data. Their data representation in the MARC bibliographic format is handled in a variety of ways. Even in the metadata world, representation of parallel writing shows some inconsistencies. It is therefore desirable to establish a new common way of representation. For this purpose, this paper discusses the class of the represented values in terms of the DCMI Abstract Model (DCAM). In the case of properties such as “Title”, it is possible to see the associated value as a “literal”, but for parallel writing, it is more appropriate to see such a value as “a sequence of words”. Accordingly, parallel writing can be represented as multiple value strings associated with a value of the class “sequence of words”. Even so, one remaining problem is that the language tags used in the value string language cannot also specify writing systems. Enumeration of the types of writing systems in various languages and registration with RFC 4646 would be required in order to express this information in DCAM value string languages.

Keywords: kanji; kana; hanja; hanzi; pinyin; metadata; MARC format; language tag; DCMI abstract model.

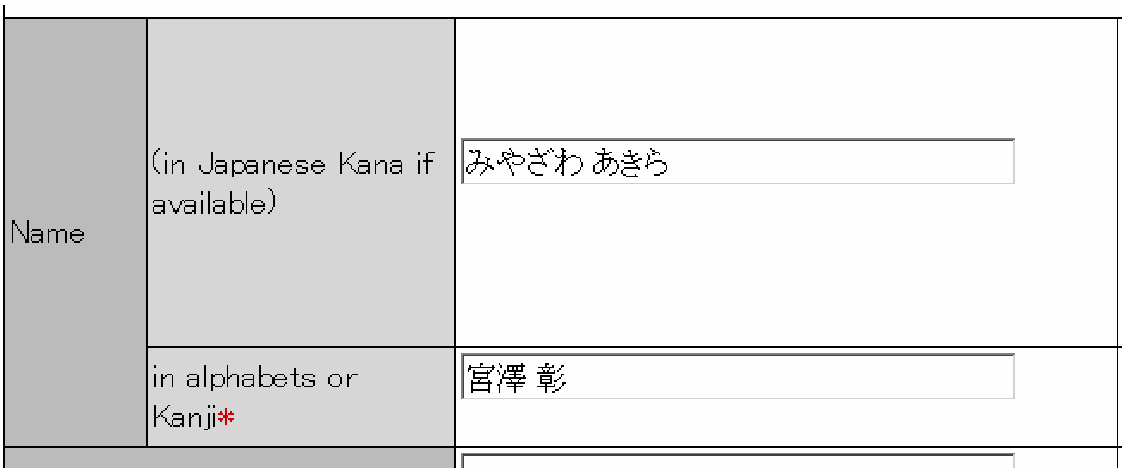
1. Introduction

FIG. 1 shows part of a typical Japanese Web form for user registration. This is a name input box, where one has to enter one’s name both in *kanji* (Chinese ideographic characters) representation and *kana* (Japanese phonetic characters) representation.

This requirement is entailed by some characteristics of the Japanese language and its writing system: (1) the modern Japanese writing system uses *kanji* together with *kana* and, occasionally, Latin alphabets; (2) A word may be represented in *kanji* or *kana* or a mixture of the two; (3) Mapping between *kanji* representation and *kana* representation is not unambiguous; i.e., a *kanji* representation may have multiple *kana* representations, and a *kana* representation may have multiple *kanji* representations. For example, my given name 彰 has *kana* representation あきら (AKIRA), but the same *kanji* 彰 can also be しょう (SHŌ), while the *kana* representation あきら (AKIRA) can be 明, 昭, 晃, 顕, or one of more than twenty *kanji* representations. 4) Personal names are represented in *kanji* in normal, everyday writing, while their collation sequence in a directory (or a catalog) follows *kana* representations.

The *kana* representation is widely known as *yomi* (reading) in Japanese, as it gives phonetic information of the word. *Kana* has two forms: one is called *katakana*, the other *hirakana*. For the purposes of this paper, the difference between the two is insignificant.

The application system processing this registration form has to handle parallel writing. Database representation of the record input from FIG. 1 is, typically, something like Table 1.



Name	(in Japanese Kana if available)	みやざわ あきら
	in alphabets or Kanji*	宮澤 彰

FIG. 1. An example of a Japanese registration form.

ID	NameInKana	NameInKanji
123456	みやざわ あきら	宮澤 彰

TABLE 1. Database representation of FIG. 1.

Parallel writing is not unique to the Japanese language. Chinese also has parallel writing of *hanzi* (Chinese ideographic characters) and *pinyin* (romanized representation of Chinese). Korean has *hanja* (Chinese ideographic characters) along with the basic script *hangul* (Korean phonetic characters). Parallel writing traditions of these three languages are different in their details, but from the viewpoint of data representation, they all have the same issues.

This paper (1) reviews how parallel writing has been handled in traditional bibliographic databases and in metadata, (2) studies the relation between DCMI Abstract Model (DCMI, 2007) (hereafter DCAM) and parallel writing, (3) identifies issues, and (4) proposes solutions.

2. MARC Formats

The first MARC format specification to cater to parallel writing was JAPAN/MARC, which was designed in 1979 (JAPAN/MARC 1981). Example 1 depicts how JAPAN/MARC handles *kanji*, *kana* and *kana* romanization.

```
251   $A 土佐日記$F 紀貫之//著
551   $A トサ ニッキ$XTosa nikki$B251
751   $A キノ，ツラユキ$XKino,Turayuki$B 紀//貫之
```

EXAMPLE 1. JAPAN/MARC.

Tag 251 subfield $A is the title and subfield $F is the responsibility statement. It is written in the usual writing system with *kanji* script. Tag 551 is the title access point field, in which subfield $A is the *kana* representation of the *yomi* of the title (Tag 251 $A) and subfield $X is the romanization of subfield $A. Similarly, Tag 751 is the author name access point field, in which subfield $A is the *kana* representation of the *yomi* of the author (Tag 251 $F) and subfield $X is the romanization of subfield $A. Tag 751 $B is the *kanji* representation of subfield $A.

(JAPAN/MARC uses some unusual punctuation marks (//) which should not be presented to end-users. This punctuation can be ignored in this context.)

China MARC (see Example 2) has a structure like that of UNIMARC (UNIMARC, 1987), but for *pinyin* representation, it has special subfields (CHINAMARC, 1996).

```
200   10   $a 西南民族教育文化溯源$Axi nan min zu jiao yu wen hua su
           yuan$f 张诗亚著$Fzhang shi ya zhu
```

EXAMPLE 2. China MARC.

Tag 200 is the title and statement of responsibility field. Subfield $a is the title and subfield $f is the statement of responsibility in *hanzi* representation. Subfield $A and subfield $F are the special subfields. They are the *pinyin* representation of subfield $a and subfield $f, respectively.

KORMARC's structure (KORMARC, 1993) is like USMARC (USMARC, 1980). It does not have special structure like JAPAN/MARC or China MARC, but still has parallel writing (see Example 3).

```
100   1    $a 케네디, 존 F.
245   10   $a 勇著의 발자취 /$dJohn F. Kennedy 著;$e 陳奉天 譯
700   1    $aKennedy, John F.
```

EXAMPLE 3. KORMARC.

Tag 100 is the author heading in *hangul* form, of which the romanized form appears as an added entry in tag 700. The title and statement of responsibility in tag 245 uses *hanja*, *hangul* and Latin alphabet writing. There is no '*hangul* only' form of the title. That is because *hanja* to *hangul* conversion is unambiguous with the Korean standard character code (KSC5601), and the system can automatically derive the *hangul* representation of tag 245 subfield $a (title) or subfield $e (translator).

USMARC (Example 4) introduced tag 880 as "Alternate Graphic Representation", with subfield $6 linking the "base" field to the "alternate" fields.

```
100   1    $6880-01$aInose, Hiroshi,$d1927-
245   10   $6880-02$aJōhō no seiki o ikite /$cInose Hiroshi cho.
880   10   $6100-01/$1$a 猪瀬 博,$d1927-
880   10   $6245-02/$1$a 情報の世紀を生きて /$c 猪瀬 博 著.
```

EXAMPLE 4. USMARC.

UNIMARC (Example 5) repeats the same field for "alternative graphic representations", with subfield $6 linking data fields and subfield $7 showing the "Alphabet/Script of Field".

```
200   1   $6a01$a 한글 자형학.
200   1   $6a01$7ba$aHan□gŭl chahyŏnghak.
```

EXAMPLE 5. UNIMARC.

These MARC examples demonstrate that, where bibliographic data is concerned, different ways of parallel representation of names and titles are common in East Asian languages. In fact, this is so not only for bibliographic data, but is commonly found in most parts of data processing in these languages.

The above examples show that there are two ways of representation. One is to coin a new data field or subfield for parallel writing. The other is to use parallel fields with some kind of attributes and/or linking information in subfields.

3. Metadata Examples

Example 6 is from a Japanese Web resource metadata record (NII, 2003).

```
<title>世界国尽</title>
<title.transcription>セカイ クニヅクシ</title.transcription>
<creator>福沢諭吉</creator>
<creator.transcription>フクザワ ユキチ</creator.transcription>
```

EXAMPLE 6. Japanese metadata.

This is a record from an old-fashioned Dublin Core based metadata, which uses a qualifier "transcription" to record *kana* representation of the title and the creator. This type of qualifier usage is controversial today. Conceptually, however, it is a simple and natural extension of the method used in Example 1 and is often observed in Japanese metadata. When the transcription type is limited and the linkage between <title> and <title.transcription> is easily found by some means, this method provides a simple solution. However, when there are many creators, and each creator has multiple transcriptions, it may be difficult to identify the linkage.

A similar type of representation is found in a Korean metadata standard. Example 7 is from "Metadata for the Human Resources in Science and Technology" (STISC, 2005). It has separate attributes for a name represented in *hangul*, romanization and *hanja*.

Person Name Korean	김삼식
Person Family Name English	Kim
Person First Name English	Sam Sik
Person Name Chinese	金三植

EXAMPLE 7. Korean metadata.

An example of another type of representation is Metadata Object Description Schema (MODS) (LC, 2006). Example 6 may be represented in MODS as shown in Example 8.

```
<titleInfo>
        <title>世界国尽</title>
        <title script="Kana">セカイ クニヅクシ</title>
</titleInfo>
<name type="personal">
        <namePart>福沢諭吉</namePart>
        <namePart script="Kana">フクザワ ユキチ</namePart>
        <role><roleTerm type="code">cre</roleTerm></role>
</name>
```

EXAMPLE 8. MODS.

This type of parallel representation is an extension of the UNIMARC (Example 5) approach. MODS has general attributes "lang", "xml:lang", "script" and "transliteration" that can be used with parallel writing. <title> and <namePart> elements in Example 8 use the "script" attribute to identify the content.

4. Parallel Writing and DCAM

Examples in the previous sections show the widespread use of parallel writing in East Asian languages. At the same time, we can see inconsistencies in how data are represented. Such a situation is not desirable from the standpoint of metadata interoperability. In this section, we discuss how parallel writing should be represented from the viewpoint of DCAM.

According to DCAM, the value associated with a property is a *resource*. For any given property, such as the DCMI metadata term "*Title*" (dc:title), one can ask with what *class* of value the property is associated. In other words, what is the range of this *property*?

It is very natural to think of that range as a literal, or character string. In this case, the three representations of titles in Example 1, "土佐日記" (Tag 251 $A), "トサ ニッキ" (Tag 551 $A) and "Tosa nikki" (Tag 551 $X) are all different *resources*. In this sense, "HAMLET" is different from "Hamlet". Some people may object that capitalization is not a significant difference, but it depends on how characters are defined. If character "A" is different from character "a", then "Hamlet" is different from "HAMLET" as a literal.

The second possible *class* is a sequence of words. A word can be represented by different scripts, or by different writing systems. In this *class*, "HAMLET" and "Hamlet" are different representations of the same word. "土佐日記" (Tag 251 $A), "トサ ニッキ" (Tag 551 $A) and "Tosa nikki" (Tag 551 $X) of Example 1 are different representations of the same sequence of words.

If the *value* associated with the property dc:title is a literal, titles in Example 1 (Japanese *kanji*, *kana* and romanization), will be represented as in Example 9, using the DC-Text syntax (Johnston, 2006).

```
Statement (
        PropertyURI ( dc:title )
        ValueString ( "土佐日記" )
)
Statement (
        PropertyURI ( dc:title )
                ValueString ( "トサ ニッキ" )
)
Statement (
        PropertyURI ( dc:title )
        ValueString ("Tosa nikki" )
)
```

EXAMPLE 9. Title as literal.

This actually says that the *resource* has three titles (like parallel titles). One cannot tell how these three *value strings* are related.

If the *value* associated with the property dc:title is a sequence of words, then Example 9 would be expressed as in Example 10, using the DC-Text syntax.

```
Statement (
  PropertyURI ( dc:title )
  ValueString ( “土佐日記” )
  ValueString ( “トサ ニッキ” )
  ValueString (“Tosa nikki” )
)
```

EXAMPLE 10. Title as sequence of words.

The question “Is the range of dc:title a character string or a sequence of words?” does not force one answer or the other. The dc:title of a manifestation in a FRBR sense may be a character string, while a dc:title of a work in FRBR sense is more likely a sequence of words.

In the case of “*Creator*”, the “literal or sequence of words” question is not directly the range of this *property*. The class of the value of *Creator* is “Agent”. If we use the *related description* with DescriptionID mechanism of DC-Text, it will be something like Example 11.

```
DescriptionSet (
  Description (
    Statement (
      PropertyURI ( dc:creator )
      DescriptionRef ( someone )
      )
    )
  Description (
    DescriptionId ( someone )
    Statement (
      PropertyURI ( dc:title )
      ValueString ( “紀貫之” )
      ValueString ( “キノ，ツラユキ” )
      ValueString ( “Kino, Turayuki” )
    )
  )
)
```

EXAMPLE 11. Creator as sequence of words in separate description.

Use of dc:title for the personal name may look strange. But when the described resource is a person, value of the property dc:title is personal name, by the definition “a name given to the resource”.

This type of multiple *related descriptions* in DCAM was introduced rather recently and is still not widely used. Name as *value string* is more commonly used. Example 12 shows parallel writing of personal names in multiple *value strings*.

```
DescriptionSet (
  Description (
    Statement (
      PropertyURI ( dc:creator )
      ValueString ( "紀貫之" )
      ValueString ( "キノ，ツラユキ" )
      ValueString ( "Kino, Turayuki" )
    )
  )
)
```

EXAMPLE 12. Creator name representations in multiple value strings.

With this representation, the distinction between "literal" as opposed to "sequence of words" does not apply, so the relation of these three *value strings* is unclear.

5. Language Tags and Writing Systems

In the examples of the previous section, I have omitted *value string language*. DCAM says that *value string language* is ISO language tag, which should be a language tag as defined in BPC 47 or RFC 4646, which renders RFC 3066 obsolete (RFC 4646, RFC 3066). Language tags were omitted from the above examples because there are no appropriate language tags to identify the three representations above. Language tags in the form of "language code-country" code are not enough to distinguish Japanese *kana*, *kanji* and Romanization, as they are all "ja-JP" (Japanese-Japan).

RFC 4646 has introduced script sub-tags. With it, one may think "ja-Hani" (Japanese-*kanji* script), "ja-Kana" (Japanese-*katakana* script) and "ja-Latn" (Japanese-Latin script) are suitable. However, it is not just a matter of the script. It is a kind of writing system difference. Example 13 is a company name.

```
Statement (
  PropertyURI ( dc:title )
  ValueString ( "IBM" )
  ValueString ( "アイ・ ビー・ エム" )
  ValueString ( "Ai bî emu" )
)
```

EXAMPLE 13. A name with writing system differences.

The first one is the usual writing system. Proper nouns in acronym form are written in the Latin script. It cannot be identified by "ja-Hani". The second one is the *yomi* in *kana*-only writing system, and the third one is *kunrei-siki* romanization of the second one. With the modified Hepburn system of romanization, which is used by the Library of Congress, it is "Ai bī emu".

In Japanese and in many other written languages, there are more than one writing system or writing system variations. For example, there may be old and new orthography, and there may be several romanization schemes. With differences in writing systems or writing system variations, one word in a language can be represented in many ways.

Sub-tags of RFC 4646 "can also be used to indicate additional language attributes of content ... indicating specific information about the dialect, writing system, or orthography" (RFC 4646, 1.).

However, there is no standard scheme for indicating a writing system. Script code is not enough to identify a writing system. To identify a writing system with RFC 4646, one would need to register sub-tags for that purpose.

6. Required Action

For the purpose of interoperability, it is desirable to establish a standard way to represent this type of parallel writing systems in DC metadata. Two courses of action should be taken to achieve this.

One is to investigate the parallel writing tradition - how parallel writing is represented in metadata. As mentioned in the previous sections, East Asian Languages use them. However, it is not clear how many other languages have parallel writing traditions of relevance to metadata.

The other course of action is to enumerate the writing system variations used for parallel writing, and to register them with RFC 4646, so that they can be used within the DCAM *value string language*.

7. Further Consideration on Parallel Writing Tradition

In this paper, I have taken the Japanese parallel writing tradition as representative of parallel writing systems in general. It must be true for most of the cases. But there are cases difficult to explain as parallel representation with different writing system. An example is a famous Japanese drama title of the 19th century, using the *rubi* layout.

Rubi is a small font *kana* representation placed to the right of the *kanji* characters in Example 14. It is not necessarily vertical writing, but can also be horizontal writing, in which case the *rubi* is placed above the main *kanji* characters. This layout is still so well used in Japanese publications that Microsoft Word supports it.

あおとぞうしはなのにしきえ
青砥稿花紅彩画

EXAMPLE 14. *Rubi* in Japanese

Rubi is usually used to show the *yomi* (reading) of the *kanji* script. Typically, it is used in educational books for children. But in this case, the *rubi* is not the usual *yomi*, but rather like a rephrasing of the title in *kanji* script. The layout suggests *yomi,* but in fact it is used to achieve some literary effect. This type of *rubi* usage is still found in cartoons, advertisements, etc. This is not the representation of words in different writing systems. It is more like parallel titles written in two languages. In a *description,* it should be recorded as separate *statements*. This means that when recording metadata, consideration should be given not only to the layout but also to the relation of the words written there.

8. Conclusion

East Asian languages have parallel writing systems. But they have different ways of representing these systems in metadata. This is not desirable for interoperability of metadata. From the viewpoint of DCAM, these are most naturally considered as different representations of a sequence of words. In this interpretation, the most suitable way is to record them as multiple value strings in a statement. But language tags for the value string are currently not enough to identify these writing system differences. To establish interoperability of such metadata, further investigation of parallel writing traditions and registration of these writing systems are needed.

To establish wider interoperability of metadata, better applicability for various languages is required. It is hoped that this paper helps to achieve wider applicability of Dublin Core metadata.

Acknowledgements

The author wishes to thank Thomas Baker for his invaluable suggestions, and Sam Oh and Inseok Song for giving indispensable information.

References

CHINAMARC. (1996). *中国机读目录格式*. 中华人民共和国文化部. WH/T0503-96.

DCMI. (2007). *DCMI Abstract Model*. Retrieved June 7, 2007, from http://dublincore.org/documents/abstract-model/.

JAPAN/MARC. (1981). *JAPAN/MARC マニュアル*. 東京. 国立 国会 図書館.

Johnston, Pete. (2006, May). DC-Text: A text syntax for Dublin Core Metadata (Working draft). Retrieved July 7, 2007, from http://dublincore.org/architecturewiki/DCText.

KORMARC. (1993). *한국 문헌자동화목록형식.* 서울. 국립중앙도서관. KSC 5867.

LC (2006). *Metadata Object Description Schema, version 3.3*. Retrieved March 30, 2007 from http://www.loc.gov/standards/mods/.

NII (2003). A metadata. Retrieved September 14, 2005, from http://ju.nii.ac.jp/oai.

RFC 3066. (2001). *Tags for the identification of language.* Retrieved March 12, 2007, from http://www.ietf.org/rfc/rfc3066.txt.

RFC 4646. (2006). *Tags for identifying languages.* Retrieved March 12, 2007, from http://www.ietf.org/rfc/rfc4646.txt.

STISC. (2005). *Standard: Metadata for the Human Resources in Science and Technology: Final draft.* 과학기술정보표준화위원회.

UNIMARC. (1987). *UNIMARC manual*. London: IFLA Universal Bibliographic Control and International MARC Programme, British Library Bibliographic Services.

USMARC. (1980). *MARC formats for bibliographic data*. Washington, D.C.: Automated Systems Office, Library of Congress.