Dublin Core の現在

杉本重雄
筑波大学・図書館情報メディア研究科

概要

Dublin Core はインターネット上のメタデータ規則として最もよく知られているといってよいであろう。この４・５年の間に Dublin Core では色々な重要な概念が明確化され，多くの分野で共通に利用されるシンプルなメタデータエレメントセットのみならず，メタデータの Semantic Interoperability を支えるためのいろいろな基本概念を与えるという役割が明確になってきたと思われる。また，Dublin Core の開発組織である Dublin Core Metadata Initiative （DCMI）もこの 4・5 年の間に大きく変化してきた。本稿では，はじめに Dublin Core の開発の歴史を振り返って，概説し，その後，重要な話題として，エレメントの再定義，DCMI Abstract Model，そして Application Profile の新しい枠組みである Singapore Framework について述べる。

キーワード

Dublin Core，Dublin Core Metadata Initiative (DCMI)，メタデータの意味的相互運用性，Application Profile，DCMI Abstract Model, Resource Description Framework (RDF)

Recent Developments in Dublin Core

Shigeo Sugimoto
Graduate School of Library, Information and Media Studies
University of Tsukuba

Abstract

Dublin Core is a well-known and well-used metadata schema among Internet users as a simple metadata element set. The Dublin Core community has clarified several important developments which supports semantic interoperability of metadata. Dublin Core Metadata Initiative (DCMI) has made significant progress in these four to five years, e.g. replication of the 15 Simple Dublin Core elements, establishment of DCMI Abstract Model, and development of Singapore Framework – new definition of application profile. This article is aimed to understand what is the Dublin Core of today – activities and achievements of DCMI. This article first reviews the history of the development of Dublin Core and then describes these recent developments which are crucial for semantic interoperability of metadata and software which uses Dublin Core.

Keywords

Dublin Core，Dublin Core Metadata Initiative (DCMI)，Metadata Semantic Interoperability，Application Profile，DCMI Abstract Model, Resource Description Framework (RDF)

1. はじめに

Dublin Core[1]は，インターネット上のメタデータのためのエレメントセットとして非常に広く認知されており，また実際に利用されてもいる。このことは疑えないが，その一方，はじめに 15 の基本要素からなるシンプルなメタデータ規則として受け入れられたがために，この 5・6 年の間に，相互運用可能なメタデータの基盤としての Dublin Core の開発が進んだにも関わらず，以前からの理解がそのままになってしまっているようにも見受けられる。この論文は，これまでの Dublin Core の開発の歴史と最近の Dublin Core の重要な話題を並べることで，Dublin Core に関する理解を深めるための解説として書いたものである。Dublin Core は出発当初から Semantic Interoperability ということをうたっていた。Dublin Core の基本モデルを定義した Dublin Core Abstract Model や Application Framework は，メタデータの相互運用性に関して Dublin Core が作り出した非常に重要な概念であり，かつ，相互運用を意識したメタデータの実現方式の基礎を与えるものである。そのため，本稿では Dublin Core が生み出した概念や，それらの考え方についての理解を深めることを目的として，現在の Dublin Core を解説する。

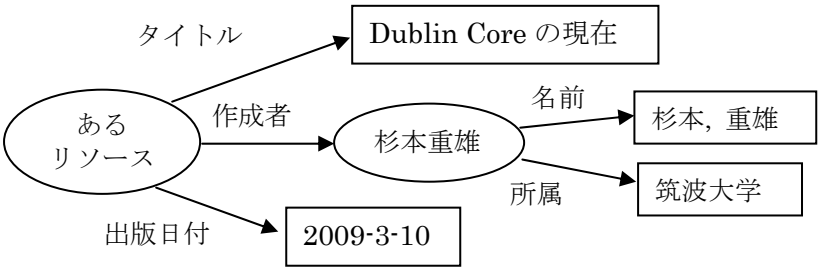
以下，第 2 章では Dublin Core の開発の歴史について記述要素集合（エレメントセット）やそれに関連する色々な概念と，Dublin Core の開発組織である Dublin Core Metadata Initiative（DCMI）の両面から述べる。第 3 章では，現在の Dublin Core が持つメタデータの記述要素集合について述べる。第 4 章では，Dublin Core の基礎モデルである Dublin Core Abstract Model について，第 5 章では，Abstract Model の開発とともに作られた Application Profile に関する新しい概念である Singapore Framework について述べる。

2. Dublin Core の歴史と基礎概念

2.1 メタデータの基礎概念

メタデータとは，一般的に，「データに関するデータ（Data about Data)」あるいは「データに関する構造化されたデータ（Structured Data about Data)」と定義される。言い換えると，メタデータとは，記述対象となる情報資源に関して，決められた属性についてその属性値を書き表したものである。図 1 は，Web 上でのメタデータ表現の標準である Resource Description Framework[2]のデータモデルを用いて表した，属性と属性値によるメタデータの表現の例である。

メタデータの記述規則は，記述対象である情報資源に対して，どのような属性について記述するか（すなわち属性の集合）と，属性の値をどのように記述するか（すなわち属性値の型・クラス）を決めたも



楕円はリソース，長方形はリテラルを表す。矢印はリソースが持つ属性を表す。
RDF では｛リソース，属性，属性値｝の三つ組みの集まりでメタデータを表現する。

図 1　RDF のデータモデル

のである。これに加えて，記述対象からどのようにして値を抽出するかについての基準も必要となる。ただし，本稿でメタデータの記述規則と呼ぶ場合は，属性の集合と属性値の型・クラスの定義，それに加えて，属性毎の記述上の要件（必須，省略可といった条件）のことを意味する[3]。また，本稿では，これをメタデータスキーマと呼び，属性を表す語（term）のことを記述項目あるいはエレメントと呼ぶ。現実のメタデータ作成には，メタデータを具体的な表現形式を決めることとメタデータ記述のための値の抽出基準を決めることが必要である。

2.2 Dublin Core の歴史の概略

Dublin Core の開発の歴史は，メタデータの記述規則を与える記述要素集合の開発とメタデータ記述に関するいろいろな概念の明確化の歴史と，こうした成果物を作り出してきた組織である Dublin Core Metadata Initiative（DCMI）の歴史の両面から見なければならない。ここでは，まず成果物であるメタデータ記述要素集合と関連概念の開発の歴史を概観し，その後で DCMI の歴史を概観する。

なお，記述要素集合のことを Dublin Core Metadata Element Set（DCMES あるいは DCES）と呼ぶことがあるが，Simple Dublin Core として定義された基本 15 エレメントセット[4]と現在 DCMI が定義しているメタデータの語彙（DCMI Metadata Terms[5]）との区別がつきにくいので，本稿では特別な場合を除き DCMES といった表現はしないことにする。

2.2.1 メタデータ規則としての Dublin Core の開発の歴史

(1) 開発初期　– 15 エレメントの標準化まで

Dublin Core の開発は 1995 年春のアメリカオハイオ州ダブリン(Dublin)でのワークショップから始まった。インターネット上での多様な情報資源（Resource）の発見のためのメタデータとして，開発のはじめは 13 個の記述項目として提案され，すぐに 15 項目になって標準化された。多様な分野の多様なリソースに対する共通のメタデータ記述規則を作ろうとすると，ごく大雑把に言って，何でもかんでも取り入れた巨大な規則（最小公倍数的メタデータ）とするか，共通部分だけを取り入れたシンプルな規則（最大公約数的メタデータ）とするかのいずれかであろう。Dublin Core は後者の方法をとったものである。

なお，開発が始まった当時は，記述対象を情報資源（Resource，リソース）という呼び方はせず，Document Like Object (DLO，文書様の実体)と呼んでいた。「文書のようなもの」と限定しなくなったことは，インターネット上に提供されるものの多様さを表してもいると理解できる。

1996 年にイギリスのウォーリック（Warwick）で開催された第 2 回ワークショップでは Warwick Framework と呼ぶ概念が提案された[6]。これは，複数のメタデータ基準を用いてメタデータを記述する際の枠組みを示したものであり，各メタデータ基準で書かれたメタデータの Package を一つのコンテナに入れるモデル（Container Model）が提案された（図 2）。Warwick Framework は，応用目的に合った詳細な記述能力を持つメタデータ基準と，一般的な内容の記述に向いた Dublin Core を組み合わせてメタデータを記述するための基本概念を与えたものと理解できる。

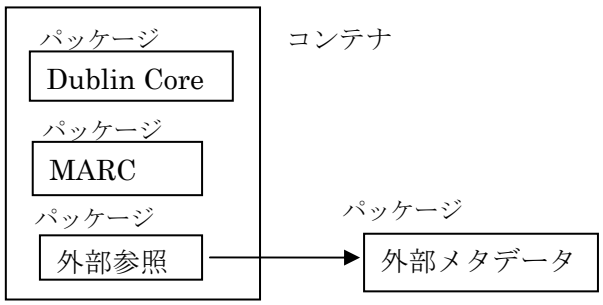


図 2　Warwick Framework のコンテナモデ

World Wide Web コンソーシアムで始められた Semantic Web の活動の中で開発された，メタデータ記述のための枠組みである Resource Description Framework（RDF）は，当然のことではあるが，特定の応用目的のメタデータ規則を指向したものではなく，任意の規則に対応するように定義されている。そこでは，別々の規則に含まれるメタデータ記述要素を組み合わせて表現できるようにモデルが作り上げられている。これは，Warwick Framework のコンテナモデルと同じ考え方と見ることができる。

Dublin Core の初期の議論の中には，あくまでも単純さを中心とするか，記述性を重視してある程度の構造を表すべきかという議論があった。たとえば，個人を表す際に，名前，所属，連絡先を 1 つの構造を持つデータとして表現することはごく一般的である。こうした構造を持つ表現を Dublin Core の中にいれるかどうかに関する議論が，1997 年春にオーストラリアのキャンベラで開催された第 4 回ワークショップでなされた。当時は，目的に合った詳細なメタデータの表現に対する要求にどのように応えるかという視点から非常に熱い議論もなされたが，その後，RDF の開発が進んだことで，RDF のデータモデルが明らかになると，Dublin Core では構造表現に関する定義はしないという合意がなされた。

1997 年秋にフィンランドのヘルシンキで開かれた第 5 回ワークショップでは，15 の基本記述要素（基本エレメント）と，それに「どのエレメントも繰返し可能であり，かつ省略可能である」という要件をあわせた Simple Dublin Core を標準とする基本的合意がなされた。Simple Dublin Core は 1998 年秋のワシントン DC でのワークショップを経て，IETF，CEN（ヨーロッパ）での標準化が進められ 2001 年には ISO での標準(ISO 15836)となった。ISO 15836 は JIS 化されて，JIS X 0836 となった。

(2) Qualified Dublin Core の導入，そして更なる概念の明確化

Simple Dublin Core の議論が進められている間にも，基本エレメントでは表すことのできない詳細度の記述をどうするかというメタデータ記述の詳細化について，以下の 2 つの異なる視点から議論された。

(i) 属性の意味の詳細化

たとえば，Simple Dublin Core の「日付」という属性には，出版した日付，受け付けた日付，有効期限の日付というように，いろいろな種類の日付を考えることができる。こうした出版日付や受付日付は，日付という属性の意味を詳細化したものであり，逆に，日付は出版日付や受付日付を一般化したものである。日付属性のままでは意味が一般的に過ぎて使いづらい，かといって意味の詳細化は領域を越えた相互運用性を阻害する可能性がある，といった面からの議論がなされた。

(ii) 属性値の記述形式の指定

日付の書き方には，平成 21 年 3 月 10 日，2009-3-10，10/03/09 というように色々な方法がある。日付がどのような標準規格に基づいて書かれているのかがわかっていることで，属性値が明確化でき，コンピュータによる処理も行いやすくなる。

こうした議論は 2000 年前後に進められ，2001 年には詳細化のための最初の属性（記述項目）と属性値の符号化の限定子が導入された。こうした限定子付きの Dublin Core を Qualified Dublin Core と呼んだ。

2000 年の 7 月には Qualified Dublin Core の最初のエレメントセットが決められた。その後も，エレメントセットの維持管理作業は現在に至るまで続けられてきている。この維持管理作業の中では，(a)新しい基本エレメント（属性）の導入，(b)既存の属性の意味を詳細化する限定子の導入，(c)属性値の記述形式を指定する限定子の認定を行った。たとえば，情報資源の想定利用者を表す基本エレメント audience が導入された。新規エレメントの導入の原則は，そのエレメントが領域にまたがって有用であるかどうかということである。また(a)と(b)に関しては，限定子を取り除けばもとの記述に戻せることという Dumb-down（ダムダウン）原則を基礎にしていた。

Qualified Dublin Core が導入されたとき，詳細化された属性の表現を既存の属性の意味を詳細化するための形容詞として導入した。たとえば，日付（Date）エレメントに対する作成日付（Date Created）のように，形容詞部分だけからなるエレメントとして導入された。しかし，その後，RDF Schema に基づくエレメントの定義が進み，「詳細化限定子」はなくなり，意味の詳細化された属性としての定義が明確になるとともに，形容詞部分を含む名詞として属性を定義するようになった。しかしながら，当初導入され形容詞として名づけられた属性は実際上の理由からそのまま残されている。また，こうした名前から開発の過程をうかがい知ることもできる。同様に，符号化の形式を表す限定子も値の型を表す要素として定義されることになった。このようなエレメントの再定義作業の議論が進んだことにより，現在では Qualified Dublin Core ということばは使われなくなっている。

(3) Application Profile の概念の明確化

Dublin Core の開発の中では，すでに Warwick Framework の段階においていくつかの異なるメタデータ規則を組み合わせた記述に関する議論がなされている。Warwick Framework は，Dublin Core のエレメントがカバーする範囲を超えて，目的毎，分野毎に特化した記述を行うための仕組みの基礎を与えている。RDF を基礎としたメタデータの捉え方が明確になっていくとともに，Dublin Core のエレメントと他のメタデータ規則で作られたエレメントを組み合わせてメタデータを記述するための Application Profile という概念が提案された[7]。

メタデータスキーマは，以下の要素からなる

(i) 属性，属性値の型・クラスを表す語（メタデータボキャブラリ）

(ii) 属性毎の記述要件（必須，省略可能性，繰返し要件，属性値の型等）

(iii) 具体的な記述形式の定義

従来のメタデータ規則では，これらを一つの完結したシステムとして定義するために，他の規則で同じような意味を持つ属性や属性値の型が定義されているとしても，独自に定義してきた。すなわち，規則毎に独自のメタデータのボキャブラリを作ってきた。これに対して Dublin Core の Application Profile の考え方は，既存のメタデータ規則から適切な属性，属性値型を選んで利用し，必要に応じて独自の語を作って，目的に応じたメタデータボキャブラリを構成し，メタデータスキーマを作ることである。（図 3 に Application Profile の概念を示す。）独自の語を定義する際にも，既存の属性や属性値型を詳細化することを勧めている。すなわち，できるだけ「アリモノ」を使うことを推奨する考え方である。

従来のメタデータ規則の考え方に基づくと，異なるスキーマで作られたメタデータの間での相互運用性を得るには，属性や属性値型の間でのマッピング（Crosswalk）を必ず作らねばならなかったのに対して，Application Profile の考え方に基づくことで，相互運用のための異なるメタデータスキーマの間での関係付けを行いやすくしているのが特徴である。

2.2.2 Dublin Core Metadata Initiative（DCMI）

現在の Dublin Core Metadata Initiative (DCMI)は Dublin Core の開発，維持管理を行っている組織である。しかしながら，開発の当初からそうした組織がきちんと作られていたわけではなく，いわばボランティアによるプロジェクトとして活動してきたものである。以下では，簡単に DCMI の歴史をたどってみたい。

(1) 開発の初期（1998 年ごろまで）

第 1 回のワークショップを開催した OCLC は Dublin Core をずっと開発を支えてきた。OCLC

Research が Dublin Core 開発の中心となり，OCLC Research の Stuart L. Weibel 博士は開発の最初からの 7・8 年をずっと引っ張ってきた。また，イギリスの UKOLN は現在に至るまで大きな貢献をしてきている。開発初期の段階では World Wide Web コンソーシアム(W3C)からの参加もあり，RDF の開発と Dublin Core の開発は互いに影響をし合っていた。

このころの開発作業はワークショップによる議論とメーリングリストを通じた議論によって進められた。参加者は，企業や組織の代表というわけではなく，草の根専門家としてメタデータ標準の開発に寄与した。

(2) 標準化の進展（2003 年ごろまで）

最初の開発が一段落し，標準化を進めるに当たって，Dublin Core Metadata Initiative をプロジェクトから，標準の維持管理組織として独り立ちさせる必要が出てきた。そのため組織構成を明確化し，Executive Director，Advisory Board，Usage Board，ワーキンググループからなる組織が整備されていった。Executve Director には Weibel 氏がつき，ワーキンググループのリーダーやメタデータの専門家からなる Advisory Board を作った。さらに，前述の Qualified Dublin Core の開発に当たって，新しいエレメントの導入を決めるための Usage Board が作られた。Usage Board は 10 名弱の専門家からなるグループで Thomas Baker 博士が現在に至るまでリーダーを務めている。Usage Board が認定した語は DCMI Metadata Terms に含められることになる。DCMI では，定義したメタデータ記述要素（term）の定義を RDF Schema で表現し，ネットワーク上で蓄積するサービス DCMI Metadata Registry を開発した。なお，DCMI Metadata Registry は OCLC で運営されてきたが，DCMI の独立の過程で，OCLC から筑波大学図書館情報メディア研究科知的コミュニティ基盤研究センターに移すことになり，現在は DCMI との合意の下に同センターにおいて運営している[8]。

また，初期段階では標準の開発のための議論が中心であったので，議論と合意形成のためのワークショップが開催されてきた。しかしながら，開発の進展とともに新しい研究成果や実践の報告と情報共有の場が必要となってきたために論文発表，グループに分かれた議論，そして新たな参加者のための講習の場を備えた国際会議として編成しなおすことにし，2001 年の東京での会議を第 1 回として現在に至っている。また，会議での発表論文は DCMI のサイトで提供されている。

(3) 組織としての独立

2000 年代に入り, Dublin Core 標準の維持管理組織として独立した組織となることがより強く求められるようになった。DCMI では，国を代表する組織を Local Affiliate とし，Local Affiliate が DCMI を支える資金を提供するというモデルを採用して組織としての独立を進めてきた。これに加えて企業等の Parner からの支援も得ている。本稿を執筆時点で Local Affiliate となっている国はフィンランド（国

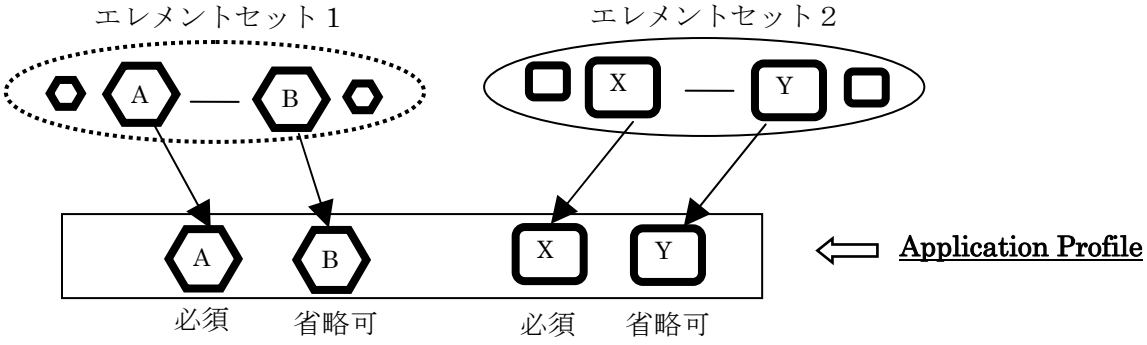


図３ Application Profile の概念図

立図書館)，イギリス（MLA と JISC)，ニュージーランド（国立図書館，国立公文書館，State Services Comission)，シンガポール（国立図書館)，韓国（国立図書館）である。2007 年には，OCLC から独立すること，組織としての登録はシンガポールで行うことを決め，2008 年になって実際に独立した。

シンガポールでの独立組織としての登録は，シンガポール国立図書館の協力の下で行われた。また，国際会議の発表論文を含む Web サイトの管理は韓国国立図書館がおこなっている。このように，DCMI Metadata Registry も加え，DCMI を支える基盤は現在東・東南アジアに置かれている。

DCMI の財政面を支える各国の Local Affiliate は，各国の事情に応じて構成されている。フィンランド，韓国，シンガポールは国立図書館を中心としているが，イギリスのようにミュージアム，図書館，アーカイブズの連携組織 MLA と高等教育の情報環境を支える組織 JISC の連携によるもの，ニュージーランドのように国立図書館，国立公文書館，そして行政サービスを束ねる組織の連携によるものもある。残念ながら我が国にはまだできていない。DCMI はインターネット上でのメタデータの相互運用の基盤を支えるための重要な活動であり，国内での理解と利用の促進も含めた組織として作る必要があると強く感じる。それには，Dublin Core のより深い理解をより多くのコミュニティに広げなければならないと思う。

3. Dublin Core の基本 15 エレメントの再定義

3.1 Dublin Core メタデータエレメントのいくつかの基本的問題

Simple Dublin Core の国際標準化を進めることに合意した 1998 年のワークショップにおいて Creator，Contributor そして Publisher の 3 エレメントをひとまとめにした Agent エレメントを導入することが提案された。これは，これら 3 エレメントの間の意味の違いが必ずしも明確ではないことが基本的な理由である。このときは，Simple Dublin Core の 15 エレメントがすでに広く流通していることから，この提案がエレメントセットの定義には反映されることはなかった。これ以外にも，Source エレメントが Relation エレメントの意味を詳細化したものであることは明らかで，標準としての Dublin Core Metadata Element Set がこうした問題を持つことは広く認識されていた。

DCMI メタデータレジストリに格納されている Dublin Core のメタデータエレメントの形式的な定義は RDF Schema で行われている。この定義において Simple Dublin Core の 15 エレメントの定義に用いられる名前空間（namespace）とそれ以外を定義で用いられる名前空間は異なっていた。ひとつの名前空間にするほうが管理上は好都合であるが，これも歴史的な理由から統合がなされないままになっていた。後述するように，最近になってになってこうした課題を解決する取組が行われた。

3.2 基本 15 エレメントの再定義

現在, DCMI のサイトを見ると二つの文書“Dublin Core Metadata Element Set, version 1.1”[4] と“DCMI Metadata Terms”[5]がある。前者は基本 15 エレメントだけを定義しており，後者は基本 15 エレメントを含むすべての語を定義している。後者の中を見ると，二つの名前空間 http://purl.org/dc/terms/と http://purl.org/dc/elements/1.1/の中にエレメント（属性，属性値の型など）が定義されていることがわかる。（以下では，名前空間 http://purl.org/dc/terms/を dcterms，名前空間 http://purl.org/dc/elements/1.1/を dc と呼ぶ。）

表 2 に DCMI Metadata Terms で定義されている語の一覧を示す。この表から，基本 15 エレメントに含まれるエレメントについては, 両方の名前空間に現れていることが理解できる。表 3 と 4 に dcterms および dc の中の creator の定義を示す。これを見ると，古くからあるエレメントの定義を与える dc の

表１　DCMI Metadta Terms ([5]より)

Properties in the /terms/ namespace	abstract, accessRights, accrualMethod, accrualPeriodicity, accrualPolicy, alternative, audience, available, bibliographicCitation, conformsTo, contributor, coverage, created, creator, date, dateAccepted, dateCopyrighted, dateSubmitted, description, educationLevel, extent, format, hasFormat, hasPart, hasVersion, identifier, instructionalMethod, isFormatOf, isPartOf, isReferencedBy, isReplacedBy, isRequiredBy, issued, isVersionOf, language, license, mediator, medium, modified, provenance, publisher, references, relation, replaces, requires, rights, rightsHolder, source, spatial, subject, tableOfContents, temporal, title, type, valid
Properties in the legacy /elements/1.1/ namespace	contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title, type
Vocabulary Encoding Schemes	DCMIType, DDC, IMT, LCC, LCSH, MESH, NLM, TGN, UDC
Syntax Encoding Schemes	Box, ISO3166, ISO639-2, ISO639-3, Period, Point, RFC1766, RFC3066, RFC4646, URI, W3CDTF
Classes	Agent, AgentClass, BibliographicResource, FileFormat, Frequency, Jurisdiction, LicenseDocument, LinguisticSystem, Location, LocationPeriodOrJurisdiction, MediaType, MediaTypeOrExtent, MethodOfAccrual, MethodOfInstruction, PeriodOfTime, PhysicalMedium, PhysicalResource, Policy, ProvenanceStatement, RightsStatement, SizeOrDuration, Standard

表２　名前空間 http://purl.org/dc/terms/中の creator の定義([5]より)

URI:	http://purl.org/dc/terms/creator
Label:	Creator
Definition:	An entity primarily responsible for making the resource.
Comment:	Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
Type of Term:	Property
Refines:	http://purl.org/dc/elements/1.1/creator
Refines:	http://purl.org/dc/terms/contributor
Has Range:	http://purl.org/dc/terms/Agent
Version:	http://dublincore.org/usage/terms/history/#creatorT-001

表３　名前空間 http://purl.org/dc/elements/1.1/中の creator の定義([4]より)

URI:	http://purl.org/dc/elements/1.1/creator
Label:	Creator
Definition:	An entity primarily responsible for making the resource.
Comment:	Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
Type of Term:	Property
Version:	http://dublincore.org/usage/terms/history/#creator-006
Note:	A second property with the same name as this property has been declared in the dcterms: namespace (http://purl.org/dc/terms/). See the Introduction to the document "DCMI Metadata Terms" (http://dublincore.org/documents/dcmi-terms/) for an explanation.

中の creator エレメントが Simple Dublin Core のエレメントの定義から変更されていないのに対し，dcterms 中のエレメントは，dc の creator エレメントと dcterms の contributor（寄与者）エレメントの両方に対して refines の関係を持つ，すなわち両者それぞれの詳細化エレメントとして定義されていることがわかる。加えて，range として Agent クラス（http://purl.org/dc/terms/Agent）を持つことが

定義されている。こうした名前空間を変えた同名のエレメントの定義は，基本 15 エレメントの他のものについても同様である。このような定義の違いから，dc に定義されたエレメントが Simple Dublin Core として定義されたときからのものであり，legacy データのための定義を保っているのに対し，dcterms の方のエレメントはこれまでに明らかになってきた問題を解決する形で再定義したものであることがわかる。このことは，dc には“legacy”という修飾語がつけられていることからも理解できる。

4. Abstract Model

DCMI Abstract Model（DCAM）は，Dublin Core で定義されたメタデータの構造とそれに含まれるいろいろな概念の意味定義を与える形式モデルである。UML の記法を利用して表した抽象度の高い形式的なモデルとして定義されているので，こうした記述に慣れない人にとっては難解に見えるかもしれないが，Dublin Core の定義を明確に伝えることができるため Dublin Core に基づくソフトウェアの設計者や開発者にとっては非常に重要な役割を持つものである。

DCMI Abstract Model[9]]は 2005 年 3 月に推奨となり，その後 2007 年 6 月に改訂されて現在に至っている。この文書は Dublin Core メタデータの基本モデルを定義するものであり，Resource Description Framework をベースにしている。この文書の中では，以下の 3 つのモデルを定義している。

・Resource Model：リソースの記述がどのような要素から成り立っているかを示す。

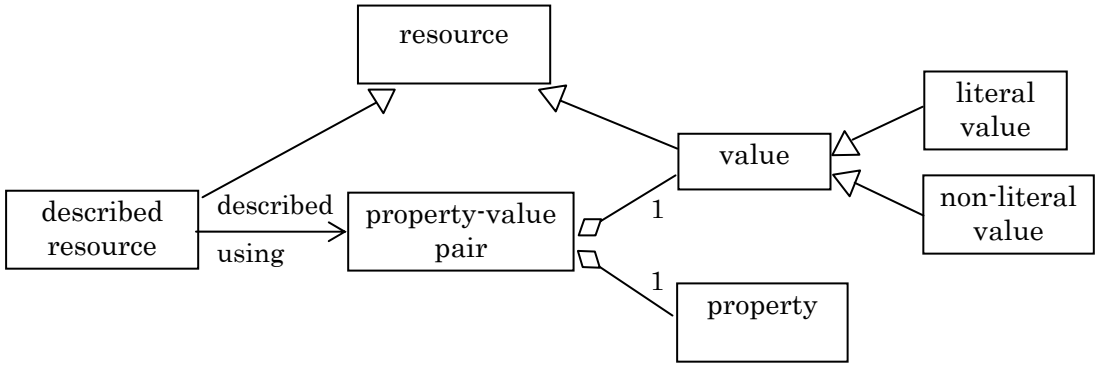
・Description Set Model：メタデータがどのような要素から成り立っているかを示す。

・Vocabulary Model：Dublin Core メタデータの語彙に含まれる語の性質とそれらの間の関係を示す。

図 4，5，6 にこの 3 モデルを示す。なお，これらの図では UML のクラス図が用いられている。以下は図の大雑把な説明である。

Resource Model は，記述対象となるリソース（described resource）は，属性と属性値の対の集まりで表されることを示している。

Description Set Model は，何らかの表現形式であらわされたメタデータの実体（record）が 1 個の description Set でできること，そして一つの Description Set には 1 個以上の description が含まれる。



contains, has-a 関係

数字は要素数，区間（例えば，0..n）は要素数の上限と下限を表す。
例：property value pair は value を 1 つ持つ
例：description set は一つ以上の description からなる（図 x x x）

is, is-a 関係

例：described resource は resource である。

ラベルで表される関係

例：described resource は。Property value pair を用いて表される(described using)。

図 4　Resource Model

ただし，ひとつの description は記述対象のリソースへの関連付けと一つ以上の文（statement）からなる。ひとつの文は属性と属性値からできており，Resource Model の property-value pair に対応する。属性値は，直接文字列で表現されるリテラルの場合（literal value surrogate）と非リテラル（non-literal value surrogate）の場合がある。非リテラルの場合には，統制語彙で決められた値を表す場合のほかは何らかのリソースを表すことになる。なお，Description Set については次節に例を示している。

Vocabulary Model は，メタデータ記述に用いられる語彙を表したもので，語彙に含まれる語（term）は属性（property）とクラス（class），符号化の構文を表すもの（syntax encoding scheme），あるいは統制語彙を表すもの（vocabulary encoding scheme）のいずれかである。また，属性は，その domain

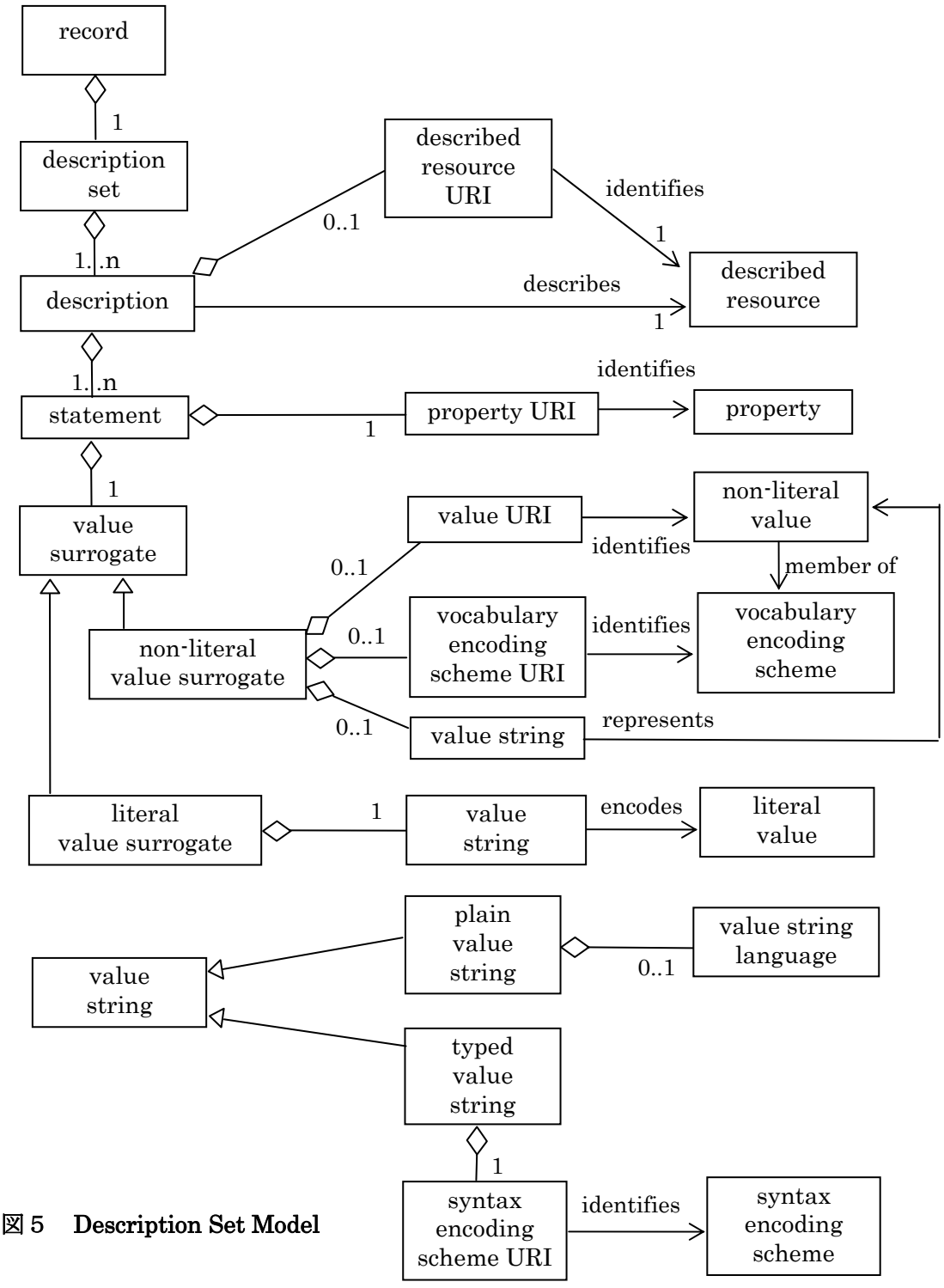


図５　Description Set Model

とrangeとなる値のクラスを与えられることがある。

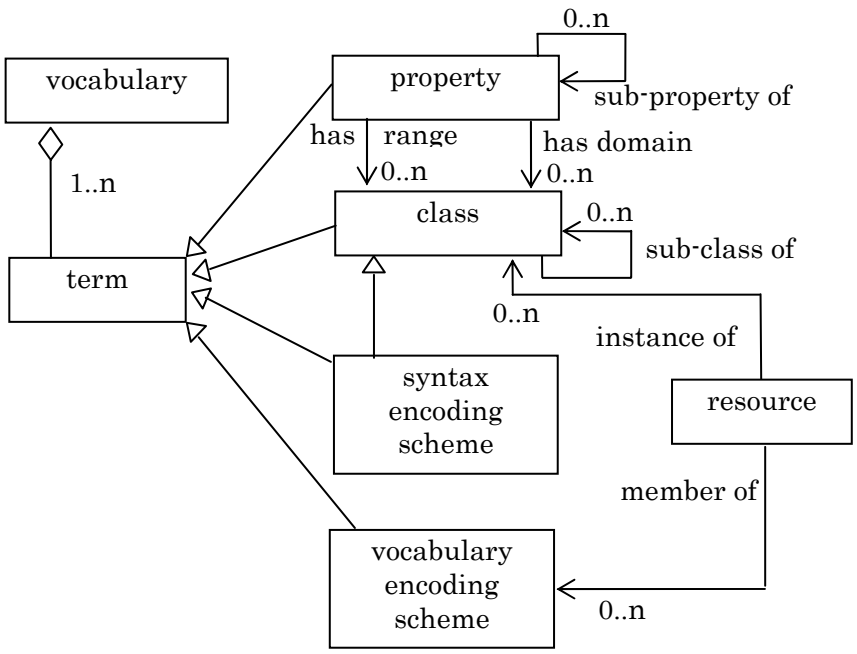


図 6 Vocabulary Model

5. Application Profile の定義

Application Profile が重要な概念であることの認識は以前からある。たとえば，Library Application Profile が作られてきた。その一方，Application Profile に関するきちんとした定義を DCMI として作ることはされてこなかった。2007 年にシンガポールで開催された国際会議では，前章に示した Abstract Model の説明の中で Singapore Framework と呼ぶ Application Profile のフレームワークが示され，それに続いて文書化もなされた[10]。

Singapore Framework は以下に示すように 5 つの要素からなっている。また，図 7 に概念図を示す。

・Functional Requirements（必須）: Application Profile によって定義されるシステムやサービスにおいて，実現することが必要とされる機能を定義する。不要な機能について述べることも含まれる。

・Domain Model（必須）: Application Profile によって定義されるシステムやサービスに含まれる基本的な実体とその実体間の関係を定義する。

・Description Set Profile（必須）: Application Profile によって定義されるシステムやサービスにおい

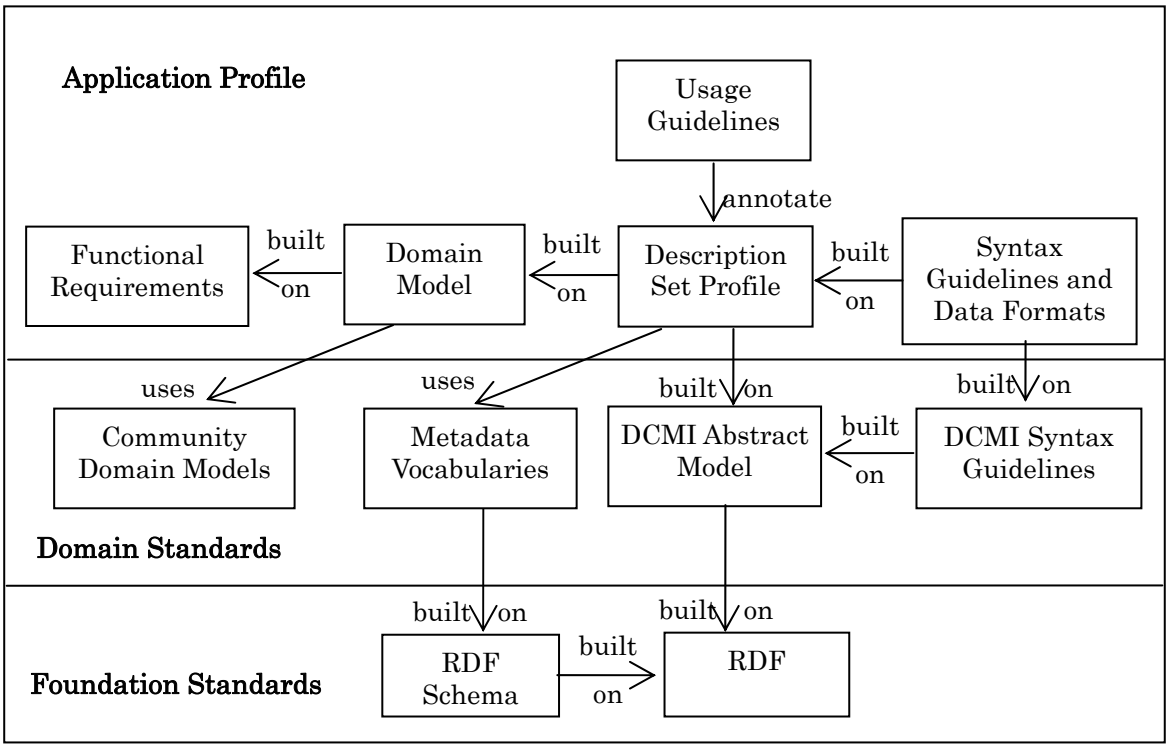


図 7 Singapore Framework

て，メタデータの実体（レコード）の構造制約を定義し，適用対象メタデータの妥当性を検証できるようにする。

・Usage Guidelines（省略可）: Application Profile 全般の応用方法や Application Profile に含まれる属性の提供方法等を示す。

・Encoding Syntax Guidelines（省略可）: Application Profile で示されるメタデータスキーマを，特定の応用システムやサービスとして実現するためのコード化方法やコード化のためのガイドラインを定義する。

従来の Application Profile は，前述したように，既存のメタデータエレメントセットから必要なエレメントを取り出し, それに記述上の要件 (必須や省略化等の構造制約) を加えて作り上げたものである。Singapore Framework はそれを一歩進め，機能要求の記述，目的のサービスの中に含まれる実体記述など, 目的のシステムやサービスを実現する上で必要な情報をまとめたものである。加えて, Description Set Profile と呼ぶ枠組みを定義し，Application Profile の中で用いるエレメントセット，構造とその上での制約などを定義している。現時点で Description Set Profile の文書[11]についてはまだ Working Draft の段階であるが，例を用いて以下にごく簡単に示す。図 8 に Description Set Profile の構成を示す。図 9 に表形式で表した簡単なメタデータスキーマの例を用いて Description Set Profile を説明する。

図 9 のスキーマでは，太線で囲んだところが入れ子になっている。したがって，このスキーマ全体では二つの Description Template があり，外側の Description Template は｛タイトル，作成者，主題，

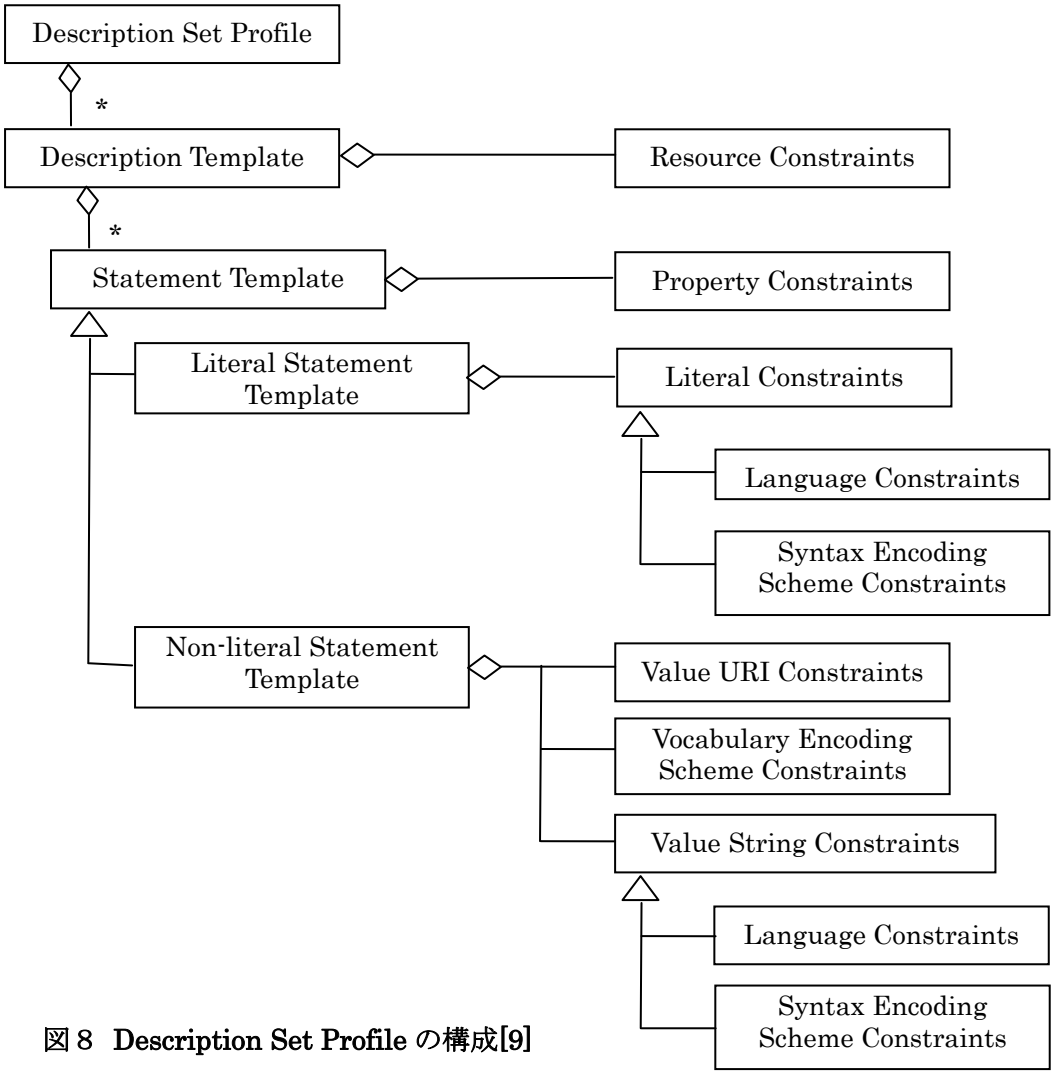


図 8　Description Set Profile の構成[9]

出版日付}，内側のものは{名前，所属，連絡先}の属性を持つ。なお，この例では陽には書かれていないが，Description Template の表現対象である文献，人あるいは組織が Resource に関する制約として与えられる。また，それぞれの属性に対して属性値の型と制約が与えられている。Statement Template は属性とそれに関わる制約属性値の組とそれに関わる制約の組からなる記述が与えられることを表している。

属性	属性値		制約
タイトル	文字列		1 個
作成者			1 個以上
	名前	文字列	1 個以上
	所属	文字列	0 個以上
	連絡先	文字列，Email アドレス	0 個以上
主題	文字列		0 個以上
出版日付	W3CDTF		1 個

属性	属性値	制約
名前	文字列	1 個以上
所属	文字列	0 個以上
連絡先	文字列，Email アドレス	0 個以上

図 9 Description Set Profile：メタデータスキーマの例

6. おわりに

本稿では, Dublin Core の開発過程を振り返り, そしていくつかの最近の重要な話題について述べた。基本 15 エレメントを定義しなおし，Legacy データの互換性を保ちつつ，懸案であったエレメント間の関係をきちんと表したこと，Application Profile の概念をきちんと整理しなおしたこと，そして Dublin Core Abstract Model として Dublin Core を支える基礎概念とそれらの間の関係を明確にしたことは，この 4・5 年の間の大きな進歩であると思う。

現在でも Dublin Core は，「15 エレメントからなるシンプルなメタデータ規則」というのが一般的な理解であると思われる。しかしながら，現在の Dublin Core には，Simple Dublin Core の 15 エレメントを含めて 70 のエレメントが登録されている。今後，新たなエレメントを加えていくことを止めたわけではないが，現在の Dublin Core の中心的な話題は Application Profile に移ってきている。Application Profile の考え方は，Dublin Core の出発点からキーワードとなっていた Semantic Interoperability（メタデータの意味的相互運用性）を得るための基本的な枠組みである。加えて，Abstract Model によって Application Profile の概念を明確化し，形式的に表現したことは，システムの開発者にとってとても有意義なことである。

本稿で述べたように Dublin Core は進化をしてきている。1995 年に始まった活動の最初の 3・4 年は Core Metadata Element Set とは何かといった概念と基本エレメントの定義，その次の 3・4 年は記述の詳細化，コミュニティ毎の要求とコミュニティ間の違いを超えた相互運用性を確保するための基本概念の開発，そして，この 4・5 年はこれまでに作られてきた概念の形式化と必要に応じた再定義，そして DCMI の組織作りが進められてきたといえる。

これまでの DCMI の成果は，相互運用性を考えて作られてきたエレメントセットと Application Profile である。実際に開発された Application Profile に基づくメタデータの相互運用とそれを支える技術の開発はまだこれからの課題である。Application Profile の考え方は「できるだけアリモノを再利用しよう」である。それには，どのようなメタデータのエレメントセットがあるのか，どのような Application Profile があるのかといった情報が容易に手に入るようにしなければならない。筆者等は，10 年ほど前から関わっている DCMI メタデータレジストリをはじめとしてメタデータスキーマレジス

トリの持つ役割は大きいと思う。ただ，それ以上にそれらを利用してスキーマを集め，共有するためのコミュニティの活動をより活発にすることが求められている。その意味では，自分の力の無さを反省するばかりである。

参考文献（下記の URL は 2009 年 2 月時点で確認）

[1] Dublin Core Metadata Initiative,　http://dublincore.org/

[2] Resource Description Framework (RDF), http://www.w3.org/RDF/

[3] Mitsuharu Nagamori, Shigeo Sugimoto, A Metadata schema framework for functional extension of metadata schema registry, Proceedings of DC-2004, 2004, http://dcpapers.dublincore.org/ojs/pubs/article/view/764/760

[4] Dublin Core Metadata Element Set, Version 1.1, 2008, http://dublincore.org/documents/dces/

[5] DCMI Usage Board, DCMI Metadata Terms, 2008, http://dublincore.org/documents/dcmi-terms/#classes-Agent

[6] Carl Lagoze, The Warwick Framework - A Container Architecture for Diverse Sets of Metadata, D-Lib Magazine, 1996, http://www.dlib.org/dlib/july96/lagoze/07lagoze.html

[7] Rachel Heery, Manjula Patel, Application profiles: mixing and matching metadata schemas, Ariadne, Issue 25, 2000, http://www.ariadne.ac.uk/issue25/app-profiles/

[8] Dublin Core Metadata Registry, http://dcmi.kc.tsukuba.ac.jp/dcregistry/

[9] Andy Powell, Mikael Nilsson, Ambjorn Naeve, Pete Johnston, Thomas Baker, DCMI Abstract Model, 2007, http://dublincore.org/documents/2007/06/04/abstract-model/

[10] Mikael Nilsson, Thomas Baker, Pete Johnston, The Singapore Framework for Dublin Core Application Profiles, 2008, http://dublincore.org/documents/2008/01/14/singapore-framework/

[11] Mikael Nilsson, Description Set Profiles: A constraint language for Dublin Core Application Profiles (Working Draft), 2008, http://dublincore.org/documents/2008/03/31/dc-dsp/