A Comparison of the Dublin Core Metadata Element Set and the Alexandria Digital Library Bucket Framework

James Frew
Donald Bren School of Environmental Science and Management
University of California, Santa Barbara
frew@bren.ucsb.edu

Greg Janée
Institute for Computational Earth System Science
University of California, Santa Barbara
gjanee@alexandria.ucsb.edu

Abstract

The Dublin Core (DC) metadata element set and the Alexandria Digital Library (ADL) bucket framework are two synoptic metadata schemes used by digital libraries. The DC element set is designed to allow humans to easily create and interpret high-level descriptive metadata, while the ADL bucket framework is designed to allow machines to easily create and search high-level metadata indexes. The two schemes are semantically similar but not completely interoperable, due primarily to the stricter representational constraints of the ADL bucket framework. A DC profile incorporating these constraints is suggested as a desirable interoperability mechanism.
Keywords*: Dublin Core, Alexandria Digital Library, bucket framework, interoperability.*

1. Introduction

The Dublin Core metadata element set and the Alexandria Digital Library "bucket" framework are two synoptic metadata schemes that strongly resemble each other, despite having been developed independently and based on quite different assumptions about the role of metadata. In this paper we examine the similarities and differences between the two schemes. In particular, we point out areas where the Dublin Core effort could benefit from the bucket framework experience.

2. Dublin Core

The Dublin Core (DC) activity was undertaken "to develop a core element set that provides adequate data for Web resource discovery and is simple for authors and content managers to create and maintain." [1]. The reference to "authors and content managers" is significant, since most metadata schemes are designed by and for expert librarians or catalogers. The DC element set was deliberately designed to be easily *populated* by non-specialists.

The current DC metadata element set [2] comprises 15 elements:

- **Title**: A name given to the resource.
- **Creator**: An entity primarily responsible for making the content of the resource.
- **Subject**: A topic of the content of the resource.
- **Description**: An account of the content of the resource.
- **Publisher**: An entity responsible for making the resource available.
- **Contributor**: An entity responsible for making contributions to the content of the resource.
- **Date**: A date of an event in the lifecycle of the resource.
- **Type**: The nature or genre of the content of the resource.
- **Format**: The physical or digital manifestation of the resource.
- **Identifier**: An unambiguous reference to the resource within a given context.
- **Source**: A reference to a resource from which the present resource is derived.
- **Language**: A language of the intellectual content of the resource.
- **Relation**: A reference to a related resource.
- **Coverage**: The extent or scope of the content of the resource.
- **Rights**: Information about rights held in and over the resource.

Some of these elements may be qualified [3] by *refinements*, which narrow the semantics of an element, and/or by *encoding schemes*, which specify domains or rules that constrain element values. (These qualifiers will be introduced where appropriate in section 4 below.)

Note that *unqualified* DC elements have no syntactic constraints at all – the element name is simply a tag associated with unstructured text. It is therefore a fundamental consequence of the DC architecture that DC metadata, while accessible to human readers or to text-based search engines, will not necessarily be accessible to more structured search mechanisms.

3. Alexandria Digital Library

In 1994 the Alexandria Digital Library (ADL) Project [4] began working towards a digital library that could both reproduce and extend the content and functionality of a traditional research map library, specifically the Map and Imagery Laboratory (MIL) at the University of California, Santa Barbara (UCSB) Davidson Library. Starting with this objective, the Project has developed four successive ADL architectures:

- a "rapid prototype" [5] system, comprising a relational database of map and imagery metadata, accessed through a desktop geographic information system (GIS);
- a "web prototype" [6] system, which replaced the stand-alone GIS with an HTTP server, generating an HTML forms-based user interface accessible via the World Wide Web;
- the "ADL-3" [7] system, which extended the HTTP server into full-fledged middleware, supporting HTTP interfaces to multiple clients, and connections to multiple catalog databases; and currently
- the Alexandria Digital Earth ProtoType (ADEPT) system, second-generation middleware that functions as a generic framework for managing, querying, and accessing georeferenced information.

As the ADL architectures became more capable, the goal of the ADL Project broadened from a "digital map library" to a network of *distributed* digital libraries of *heterogeneous georeferenced* information. "Distributed" means that libraries or their components may be spread across the Internet, as well as coexisting on a single desktop. "Heterogeneous" means that libraries may contain multiple types of digital information, including non-traditional items such as remotely sensed imagery, executable models, and multimedia instructional materials. "Georeferenced" means that, whenever possible, each item in a library is associated with one or more regions on the Earth's surface. (We refer to these regions as the item's *footprint*.)

3.1. ADL Bucket Framework

The current ADEPT architecture [8] comprises three standard abstractions, which, by virtue of being implemented identically at each ADL site, serve to bind a network of distributed libraries into an apparently homogeneous federation.

- *Interfaces* provide standard services for clients (via XML and HTTP), peer libraries (via Java RMI), and collections (via highly configurable collection drivers).
- *Reports* describe the characteristics and accessibility [9] of individual library items.
- *Indexes* implement standard search capabilities.

In the ADEPT architecture, queries are constraints on searchable indexes; query results are reports containing descriptive metadata. The *ADL bucket framework* specifies the capabilities and behavior of the searchable indexes.

The term *search bucket* [10] was coined in earlier versions of ADL to describe a metadata element with a well-defined search behavior; "bucket" denoting that the searchable element would typically be populated by the union of several underlying descriptive elements. The current bucket framework is more precise about this distinction.

An ADL bucket is a strongly typed, searchable index into a particular class of library item (*source*) metadata. The association between a bucket and a source metadata element is a *mapping*. Multiple source elements may be (and usually are) mapped into a single bucket, but this is not equivalent to simply creating a new union element, for mappings from the bucket values back to the source elements are preserved. In other words, if an operation on a bucket yields a particular value, it also yields the source metadata element from which that value was derived.

The ADL bucket framework has two components: bucket types and core buckets. *Bucket types* specify the domain of values and their associated query operators. Table 1 lists the bucket types that are implemented in the current ADEPT architecture. For example, a *spatial* bucket maps source metadata into polygons or polylines in WGS84 geographic coordinates. A spatial bucket's index supports queries expressed as contains, overlaps, or contained-in relationships between the mapped metadata values and a query-supplied box or polygon in WGS84 geographic coordinates.

The current implementation of the ADEPT architecture defines 10 *core buckets* (Table 2), that is, buckets that are deemed to be generally useful enough to be universally supported

Table 2: ADEPT core buckets		
Bucket name	**Bucket type**	**Description**
`Subject-related text`	textual	text indicative of the subject of the item, not necessarily from controlled vocabularies.
`Title`		the item's title. This bucket is a sub-bucket of Subject-related text.
`Assigned term`		subject-related terms from controlled vocabularies. This bucket is a sub-bucket of `Subject-related text`.
`Originator`		names of entities related to the origination of the item
`Geographic location`	Spatial	the subset of the Earth' surface to which the item is relevant
`Date`	Temporal	the calendar dates to which the item is relevant
`Object type`	hierarchical	ADL Object Type Thesaurus (image, map, thesis, sound recording, etc.)
`Feature type`		ADL Feature Type Thesaurus (river, mountain, park, city, etc.)
`Format`		ADL Object Format Thesaurus (loosely based on MIME)
`Identifier`	Qualified textual	names and codes that function as unique identifiers

Since bucket mappings are defined and implemented by individual collections, core buckets are simply a convention; nonetheless, the current set of core buckets comprises the logical point of tangency between the ADL and Dublin Core metadata models. This apparent semantic equivalence, however, is somewhat misleading – the specific search capabilities of the ADL buckets interact with the much broader semantics of the Dublin Core elements in interesting and occasionally incompatible ways, as elaborated in the next section.

Table 1: ADEPT bucket types			
bucket type	**value type**	**Query**	
		term type	**operators**
spatial	any of several types of geometric regions defined in WGS84 latitude / longitude coordinates, expressed in an ADEPT-defined syntax	WGS84 box or polygon	`contains`, `overlaps`, `is-contained-in`
temporal	(range of) calendar date(s) in ISO 8601 syntax	same as value type	
hierarchical	term from controlled vocabulary or thesaurus		`is-a`
textual	text		`contains-all-words`, `contains-any-words`, `contains-phrase`
qualified textual	text with optional associated namespace		`matches`
numeric	real number in standard scientific notation		standard relational operators

4. Comparison of ADL buckets to Dublin Core

In this section we present a detailed comparison between the buckets and bucket types that make up the ADL bucket framework, and the recommended elements, refinements, and registered encodings that comprise Dublin Core. As will be seen, there is a generally good semantic match between the two schemes, but there are numerous differences and incompatibilities at the representational and encoding levels that reflect fundamental differences in the schemes' respective intended purposes.

In the following discussion, to avoid confusion between similarly-named ADL buckets and Dublin Core elements, we prefix bucket names with "ADL:" and Dublin Core element names with "DC:" We follow the Dublin Core convention of representing element refinements with a "." separator; e.g. coverage.spatial.

4.1. ADL:geographic-locations ↔ DC:coverage.spatial

The ADL:geographic-locations bucket is semantically equivalent to the DC:coverage.spatial element refinement: both describe the geographic extent of the resource. (Actually, DC:coverage.spatial does not state that it describes specifically the resource's geographic spatial extent, but judging from the registered encoding schemes, the geographic domain appears to be its intended purpose.)

The differences between the two schemes lie in the allowable values and value representations. The ADL:geographic-locations bucket is intended to support a spatial search service, specifically one that performs range searching (employing inequality operators such as BETWEEN and OVERLAPS) over the latitude/longitude coordinate system. Thus the only allowable bucket values are those that directly support this functionality, namely, geometric regions (ADL supports spherical polygons and polylines, graticule-aligned boxes, and points) defined using latitude/longitude coordinates. Furthermore, the ADL spatial bucket type defines specific XML representations for these values.

By contrast, the DC:coverage.spatial element refinement allows a much broader range of values. As noted in section 2, Dublin Core elements are fundamentally typeless; that is, they mandate no particular data type or encoding or representation. This lack of syntactic definition is evident in this example of a DC:coverage.spatial value, taken from Dublin Core usage guidelines:

Columbus, Ohio, USA; Lat: 39 57 N Long: 082 59 W

While this value certainly adheres to the DC:coverage.spatial element's semantic definition, and is readily human-readable, it follows no well-known syntax scheme for either a qualified placename reference or a coordinate-based location definition. Because this fundamental difference between the (non-textual) ADL buckets and Dublin Core elements affects all correspondences between the two schemes, we won't mention it again and will instead focus on Dublin Core's encoding schemes, which do provide syntactic definition.

The DC:coverage.spatial element supports, as of this writing, four encoding schemes: TGN, ISO3166, Box, and Point. Even among these schemes we find significant differences from ADL's approach.

The TGN and ISO3166 encoding schemes encode spatial extent by making references to controlled vocabularies of placenames (the Getty Thesaurus of Geographic Names and the ISO 3166 list of country codes, respectively). From ADL's perspective, the fundamental issue here is the ability to automatically and unambiguously convert placename references to coordinate representations. For ISO3166 codes this does not present too much difficulty, since the list of countries is fixed and bounding box and even polygonal definitions of country borders are publicly available. TGN is more problematic since the Getty Thesaurus is a much larger, living database, meaning that the conversion can be defined only in terms of dynamic thesaurus lookups. This presents two significant complications. First, the TGN encoding scheme does not specify exactly what form the reference should take or what syntax it should use: placename, identifier, or qualified placename (with specific qualification syntax). We would expect that most users would tend to refer to a Getty Thesaurus entry by placename, and this leads to the second complication: placenames are not unique, even within the Getty Thesaurus.

The Box and Point encoding schemes present a different kind of difficulty. The two schemes exactly match the semantics of ADL's box and point geometric regions, but while the latter are defined specifically in latitude/longitude coordinates, the Dublin Core versions may be specified in any projection and coordinate system and using any units of measure.

4.2. ADL:dates ↔ DC:coverage.temporal

The ADL:dates bucket is semantically equivalent to the DC:coverage.temporal element refinement: both describe the temporal extent of the resource.

The representation-level comparison between these two schemes is entirely analogous to the previous comparison between ADL:geographic-locations and DC:coverage.spatial. The ADL:dates bucket supports range searching over calendar-scale time, and thus restricts acceptable values to ranges of calendar dates expressed in ISO 8601 format. By contrast, the DC:coverage.temporal element refinement supports two encoding schemes, W3CDTF and Period. W3CDTF is a restricted profile of ISO 8601, and as such is a good match for ADL's represention. But the Period encoding scheme, which describes a time interval, allows the endpoints of the interval to expressed using any scheme, not necessarily W3CDTF or ISO 8601.

4.3. ADL:titles ↔ DC:title
ADL:subject-related-text ↔
DC:title + DC:subject + DC:description
ADL:assigned-terms ↔ DC:subject
ADL:originators ↔ DC:creator + DC:publisher

With ADL's text buckets – ADL:titles, ADL:subject-related-text, ADL:assigned-terms, and ADL:originators – there are no issues of value incompatibilities, and the only differences between these buckets and their Dublin Core equivalents are in their semantic definitions.

The ADL:titles bucket is semantically equivalent to DC:title: both hold a title of the resource.

The ADL:subject-related-text bucket is intended to encompass all text related to the subject of the resource, and as such it is roughly equivalent to the union of Dublin Core elements DC:title, DC:subject, and DC:description, although the ADL:subject-related-text bucket could hold additional text, and even the content of a text-based resource if appropriate. The intention of the ADL:subject-related-text bucket is that it offer something akin to a "search over everything" or free-text search capability. To deal with cases when a search against ADL:subject-related-text returns too many results, the ADL bucket framework includes the ADL:assigned-terms bucket, by definition a sub-bucket of ADL:subject-related-text, which includes only terms from controlled vocabularies, both formally and informally defined. The ADL:assigned-terms bucket is semantically equivalent to the DC:subject element. However, the ADL:assigned-terms bucket, being a text bucket, offers text search operators ("contains-all-words", etc.) that are intended for and appropriate for text. By

contrast, several of the encoding schemes DC:subject (e.g. DDC for Dewey Decimal Classification) are alphanumeric codes, not textual words.

The ADL:originators bucket is semantically equivalent to the union of the DC:creator and DC:publisher elements and, depending on the nature of the resource, the DC:contributors element. As with ADL:subject-related-text, the definition of the ADL:originators bucket is motivated by certain desired search behavior. In this case, ADL users typically find it more convenient and sufficiently powerful to be able to search over all originator names at once as opposed to severally. (However, it should be noted that ADL offers separate mechanisms for searching over individual metadata fields that have been mapped to buckets.) This generalized search behavior is particularly appropriate for the kinds of geospatial information ADL has dealt with, where resources often have multiple originators that do not fall into neat creator/publisher distinctions.

4.4. ADL:identifiers ↔ DC:identifier

The ADL:identifiers bucket is semantically equivalent to the DC:identifier element: both describe an unambiguous reference to the resource.

The ADL:identifiers bucket allows identifier namespaces to be explicitly declared so that, for example, an identifier can be declared as residing in the "ISBN" namespace if it is an International Standard Book Number. Such declarations enhance the matching of identifiers in two ways. First, they limit accidental matches ("false positives") as in an ISBN that happens to match a telephone number because both are 10-digit hyphenated numbers. Second, the matching operation can be informed by knowledge of identifier syntax. For example, matching of ISBNs might take advantage of the fact that hyphens in ISBNs are optional and carry no semantic value. The DC:identifier element supports the URI encoding scheme, but otherwise has no mechanism for declaring namespaces.

The DC:identifier.bibliographicCitation element refinement, which references a resource by its bibliographic citation, is, strictly speaking, compatible with the ADL:identifiers bucket since ADL ultimately treats identifiers as text strings. However, the bucket's "matches" operator is really intended for code-like identifiers; users would tend to find the word-based operators offered by a text bucket ("contains-all-words", etc.) more appropriate for searching over citations. This difficulty is arguably present in Dublin Core itself, for DC:identifier.bibliographicCitation does not conform to the spirit of DC:identifier and hence is not a true element refinement.

4.5. ADL:types ↔ DC:type

The ADL:types bucket is semantically equivalent to the DC:type element: both describe the nature or genre of the resource.

In the ADL bucket framework, ADL:types is an example of a bucket whose allowable values are drawn from a specific thesaurus tied to the bucket's definition, in this case the ADL Object Type Thesaurus. We call such a bucket a hierarchical bucket because it supports automatic term expansion by exploiting inter-term thesaurus relationships. For example, a search for "images" in the ADL bucket framework will automatically return both "images" and "aerial photographs" because the latter term is a narrower (more specialized) term of the former.

The DC:type element supports the DCMIType encoding scheme that references the Dublin Core type vocabulary. The two vocabularies are largely complementary. The Dublin Core vocabulary is a shallow (one-level) list of 10 terms that attempt to provide an exhaustive categorization of resource genres. By contrast, ADL's vocabulary is a narrow, deep hierarchy focusing on making pragmatically useful distinctions between genres of geospatial materials. An excerpt:

- cartographic works
  - maps
- images
  - photographs
    - aerial photographs
  - remote-sensing images
    - aerial photographs

Vocabulary reconciliation is a well-known and still largely unsolved problem. But in this particular case, the two vocabularies are so close to being perfectly complementary that ADL is considering recasting its type thesaurus to be a direct refinement of the Dublin Core type vocabulary. This would simplify the mapping between the two.

4.6. ADL:formats ↔ DC:format

The ADL:formats bucket is semantically equivalent to the DC:format element: both describe the manifestation or representation of the resource.

As with ADL:types, the ADL:formats bucket is a hierarchical bucket tied to a particular thesaurus, in this case the ADL Object Format Thesaurus. This thesaurus reflects a blend of three semantic concepts: accessability of the resource (online vs. offline), the physical medium for offline resources, and file format (i.e., MIME type) for online resources. A fragment of the thesaurus:

- offline
  - digital
    - CD-ROM
  - non-digital
    - paper
- online
  - image
    - jpeg
  - text
    - html

Dublin Core defines no vocabulary associated with DC:format. The DC:format.medium element refinement

describe the physical medium of the resource as a freeform string. The IMT encoding scheme defines format by MIME type.

5. Mapping Between ADL Buckets and Dublin Core

In this section we consider the issues involved in mapping metadata, in an automated fashion, between Dublin Core and the ADL bucket framework.

As we saw in the preceding section, the ADL bucket framework closely matches Dublin Core at a semantic level. With only a few exceptions, the differences between the two schemes can largely be characterized as ADL placing greater restrictions on allowable metadata values and representations. Thus, to map from Dublin Core to the ADL buckets one must filter out Dublin Core metadata values that have unknown or unrecognized semantics or encoding schemes, and convert the remaining values to the ADL-mandated representations.

DC:coverage.spatial can be mapped to ADL:geographic-locations if an encoding scheme is specified. For the TGN encoding scheme, the TGN entry being referenced must be specified by a TGI identifier or by an unambiguous (possibly qualified) placename. For the Box and Point encoding schemes, the coordinate system and units used to encode the location(s) must be recognized and convertible to latitude/longitude coordinates.

DC:identifier can be mapped to ADL:identifiers without any problem. If the URI encoding scheme is present, it becomes the identifier's namespace. The DC:identifier.bibliographicCitation element can be mapped as well although, as noted previously, the match-based searching supported by the ADL:identifiers bucket will not result in effective searching.

The converse mapping from the ADL bucket framework to Dublin Core is simpler, since the Dublin core elements are in general less restrictive than the ADL buckets.

The ADL:geographic-locations bucket can be directly mapped to DC:coverage.spatial: mapped boxes and points take on the Box and Point encoding schemes, respectively. The only exception is the ADL bucket supports polygons and polylines, for which there are no registered Dublin Core encoding schemes. However, these types of geometric regions can be trivially replaced by their bounding boxes, albeit with loss of fidelity.

The ADL:identifiers bucket can be directly mapped to DC:identifier. Any non-URI namespace associated with the identifier is discarded.

6. Analysis

The ADL Project and Dublin Core arrived at remarkably similar categories of metadata, even though they started from different assumptions and worked with different goals and different communities in mind.

ADL buckets are effectively search indexes, and the definition of a bucket includes the bucket's search operators, the allowable query terms, and the semantics of the search operation. From this search-oriented definition follow restrictions on the acceptable values that may be mapped to the bucket. By contrast, Dublin Core has focused on the descriptive aspects of metadata, by providing open-ended mechanisms for declaring and using arbitrary encoding schemes.

It is instructive to look at the differences between ADL buckets and Dublin Core elements in terms of *placement of burden*; i.e. the relative burdens they impose on providers versus users of metadata. ADL's approach places a greater burden on metadata *providers* to create relatively high-quality metadata that adheres to rigid specifications. Given such metadata, search services and other programmatic clients (users of ADL metadata) can make ready use of it.

By contrast, with Dublin Core's approach the burden is placed on the *users* (readers) of the metadata. Metadata creators are granted almost unlimited freedom to populate metadata elements using arbitrary syntaxes and encodings. But that freedom (which is exuberantly exercised in practice, as any reader of Dublin Core metadata can attest) places a proportionately much larger burden on programmatic readers of the metadata, particularly structured search services that rely on certain forms of highly structured metadata. Programmatic metadata readers receive little support from the Dublin Core standard, and are forced to (re)invent additional specifications, conversion functions, and heuristic algorithms to deduce encoding schemes.

The correct apportionment of these burdens in digital libraries is difficult to gauge, and perhaps there is no definitive answer. On the one hand, an argument for Dublin Core's placement of burden is that metadata is notoriously expensive to create, and thus barriers to its creation should be as low as possible. On the other hand, an argument for ADL's placement of burden is that metadata, once created, is read many times by many different clients, particularly automated clients that must be able to interpret a metadata element based solely on its definition.

7. Conclusion

There is generally a close relationship and good semantic match between the ADL bucket framework and Dublin Core elements. The largest differences and incompatibilities between the two systems arise in the handling of encoding schemes. The ADL bucket framework restricts the allowable forms of metadata in order to support the creation of enhanced (non-textual) searchable indexes over the metadata. Dublin Core allows a more open-ended approach with multiple encoding schemes and variants; the emphasis is on documenting which are in use.

We suggest that there may be value in defining an "enhanced searchability profile" of Dublin Core. Such a profile would describe a restricted set of element refinements and encoding schemes, along the lines of the ADL bucket definitions, that would support the automatic creation of common, enhanced search indexes such as range searching over space and time. Search service providers could use the profile as a means of easily and formally

declaring their metadata requirements. For metadata providers the profile would be optional, of course, but the profile could serve as a well-known compatibility level to attempt to reach, and metadata providers that adhere to the profile would know that their content would be discoverable by advanced search services.

References

[1] Weibel, S.L., and Lagoze C. (1997). An element set to support resource discovery. *International Journal on Digital Libraries, 1*(2): 176-186.

[2] ISO (2003). Information and documentation – the Dublin Core metadata element set. ISO 15836:2003(E). Retrieved May 17, 2003, from http://www.niso.org/international/SC4/n515.pdf

[3] DCMI Usage Board (2003). DCMI Grammatical Principles. Retrieved May 17, 2003, from http://dublincore.org/usage/documents/principles/

[4] Smith, T.R., Janée, G., Frew, J. and Coleman, A. (2001) The Alexandria Digital Earth Prototype System. *in* JCDL 2001: First ACM/IEEE-CS Joint Conference on Digital Libraries, Roanoke, VA, 2001, ACM Press, 118-119.

[5] Frew, J., Carver, L., Fischer, C., Goodchild, M., Larsgaard, M., Smith, T. and Zheng, Q. (1995). The Alexandria Rapid Prototype: building a digital library for spatial information. *in* 1995 ESRI International User Conference, Palm Springs, CA.

[6] Frew, J., Freeston, M., Kemp, R., Simpson, J., Smith, T., Wells, A. and Zheng, Q. (1996). The Alexandria Digital Library testbed. *D-Lib Magazine, 2* (7/8).

[7] Frew, J., Freeston, M., Freitas, N., Hill, L., Janée, G., Lovette, K., Nideffer, R., Smith, T. and Zheng, Q. (2000). The Alexandria Digital Library architecture. *International Journal on Digital Libraries, 2*(4): 259-268.

[8] Janée, G. and Frew, J. (2002). The ADEPT digital library architecture. *in* Second ACM/IEEE-CS Joint Conference on Digital Libraries, Portland, OR, ACM Press, pp. 342-350.

[9] Janée, G., Frew, J., and Valentine, D. (2003). Content access characterization in digital libraries. *in* Third ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, TX, ACM Press, in press.

[10] Frew, J., Freeston, M., Hill, L., Janee, G., Larsgaard, M. and Zheng, Q. (1999). Generic query metadata for geospatial digital libraries. *in* Meta-Data '99 Third IEEE META-DATA Conference, Bethesda, MD. Retrieved May 17, 2003, from http://www.computer.org/proceedings/meta/1999/papers/55/jfrew.htm.