Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

Object-Oriented Representation of DDC and the Number-Building Process

Mark Giguere
Steven Shadle
Jeanne Galbraith
Information Science Ph.D. Program
Rockefeller College of Public Affairs & Policy
State University of New York at Albany
135 Western Avenue
Albany, NY 12222
MG3721@ALBNYVMS.BITNET
SS1690@CNSUNIX.ALBANY.EDU
JGALBRAI@HSCLIB.HSC.SUNYSB.EDU

The Dewey Decimal Classification [DDC] rule base and the number-building process are well suited to object-oriented analysis and design. This research examines a representation of a small portion of the DDC General Schedules via the object-oriented approach. At the most superficial level, application of DDC rules results in the designation of classes of bibliographic items, which, in and of themselves, can be thought of as objects. However, the actual underlying structure of the DDC rule base can also be thought of as a systematic assembly of objects; that is to say, distinct subsets of types of DDC rules can be identified via an object-oriented analysis. Furthermore, the hierarchical force exhibited by DDC rules is easily and naturally supported as inherited characteristics represented by the hierarchical class associations between objects in an object-oriented model. Unlike a traditional relational model representation, the ability of an object-oriented model to subsume algorithmic, procedural activities (i.e., methods) within object classes (i.e., categories of DDC rules) provides for a dynamic rule repository that easily adapts to revision of the rule base due to both the modularity of the methods and to their logical and physical separation from the actual data (i.e., the text of the rules). Full-scale implementation of the prototype design presented here could be used in the development of a decision support system for entry level catalogers. This research also has implications for development of enhancements to the user interfaces of online public access catalogs [OPACS] in libraries operating under DDC.

INTRODUCTION[1]

Although object-oriented analysis and design is a relatively new approach for the design of user interfaces (e.g., Macintosh or Windows operating systems environment) and information systems (e.g., Mosaic), the software development community has been quick to embrace the new capabilities offered by this approach to systems analysis. The modularity of code resulting from such an analysis improves system adaptability as well as supporting portability across a variety of platforms. Indeed, because of these virtues, object-oriented design and analysis has been implemented in a wide range of applications including relational database management and design (Blaha et al 1988 / Frank 1987), hypermedia database design (Parsaye et al. 1989), software development environments (Izygon 1992 / Auty 1988), computer-aided instruction (Bonar et al. 1986), database query construction (Straube & Tamer 1990), system simulation (Popken 1991), and user interface design (Ewing 1986).

---

1. This work was completed in fulfillment of the team project design requirement for the core *Proseminar in Information Organization (INF 702)* at the Information Science Ph.D. Program at the State University of New York at Albany. Assistance in development of the model was provided by Professor Jagdish Gangolly. The object-oriented analysis and design methodology used here (i.e., *Object Management Technique [OMT]*) was developed at the General Electric Research and Development Center (Rumbaugh et al 1991).

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

Research by library and information science scholars has also explored the suitability of object-oriented design and analysis to the development of information systems (Bhalla 1991 / Guzev & Titov, 1991 / Hinsch 1990). Some of this research has focused on the use of the approach as a means of enhancing user instruction (Kochtanek 1987). In the realm of classification research, it has been demonstrated that the object-oriented methodology lends itself particularly well to thesaurus representation (Schmitz 1991 / Rostek & Fischer 1988). Indeed, the development of artificially intelligent, classification-oriented applications is frequently predicated upon an object-oriented style of analysis (Liu 1993). The frames upon which such systems are predicated can themselves be thought of as objects. The authors are aware of no published work, however, that explores the use of the object-oriented approach in the representation of the rule base of a classification scheme.

This paper details an object-oriented design and analysis of a representative portion of the rule base of *Dewey Decimal Classification (20th ed.)* [DDC], a classification system used to organize general library collections (Comaromi 1989). Because objects frequently have algorithmic procedures associated with them (i.e., methods), this design is also capable of dynamically representing the DDC number-building process that results from application of said rule base. In this manner, the system design presented here could serve as prototype for the development of a cataloger decision support system that would assist in the step-by-step construction of a Dewey classification number. After a brief presentation of the fundamentals of object modelling and the characteristics of hierarchy and inheritance present in such an analysis, we will demonstrate similar constructs as they are manifested in the structure of the DDC rule base. Finally, we will present our object-oriented model of a demonstrative portion of the DDC rules and discuss how the model is dynamically capable of representing the number-building process.

OBJECT MODELLING

Object-oriented design [OOD] and analysis was first formulated as a means of enhancing and streamlining the software development lifecycle. While traditional software development approaches follow a dual approach, with the development of a data structure hierarchy proceeding simultaneously with the development of a procedural hierarchy, object-oriented design subsumes both of these processes in the development of a single, unified class hierarchy. As will become apparent during the subsequent presentation of this modelling technique as applied to DDC, OOD requires an analysis of the problem space at a high level of abstraction. Grady Booch, the father of object-oriented development in the realm of software engineering, describes it as

> ...an approach to software design in which the decomposition of a system is based on the concept of an object. An object is an entity whose behavior [i.e., methods] is characterized by the actions that it suffers and that it requires of other objects (1986).

As will become apparent in subsequent examples, objects are the application-domain *concepts* that can be uniquely identified as separate entities on the basis of their inherent identity. Interactions between these conceptual entities are captured in terms of encapsulated, procedural algorithms; that is to say, the ways in which each conceptual entity can interact with others is via internal procedural algorithms. To reiterate, while objects that possess the same attributes and methods can be grouped into the same class of conceptual entities, each instance of that class of entities is inherently distinct.

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

Once an object-oriented analysis has been conducted, and the procedural algorithms associated with objects, known as methods, are encoded, the logical and physical separation of both data and procedural algorithms provides for enhanced system adaptability. As opposed to a typical relational model design, where data that is operated on by the same algorithms is frequently collocated in a single logical data structure, object-oriented design obviates this need. It is only when a particular instance of an object is invoked that data is retrieved, from potentially logically disparate and physically separated data structures, to impart the necessary characteristic to a given invocation of a unique instance of an object. In this way, the data associated with particular objects can be changed without adverse effect on that object's associated methods, and the modularity of methods allows the programmer to "...extend the expressive power of subprograms and packages by making them generic" (Booch 1986, p. 211).

In order to facilitate an understanding of the fundamentals of object-oriented design and analysis, let us examine a simple object model representing a typical, corporate personnel database: Using the OMT system of model representation, the seven classes of objects in the figure (i.e., PERSON, WORKER, MANAGER, PROJECT, COMPANY, DEPARTMENT, and PRODUCT) are contained in boxes. The label for each object occurs at the top of each box. Contained within an object *may* be data, termed *attributes*, and/or procedural algorithms, termed *methods* or

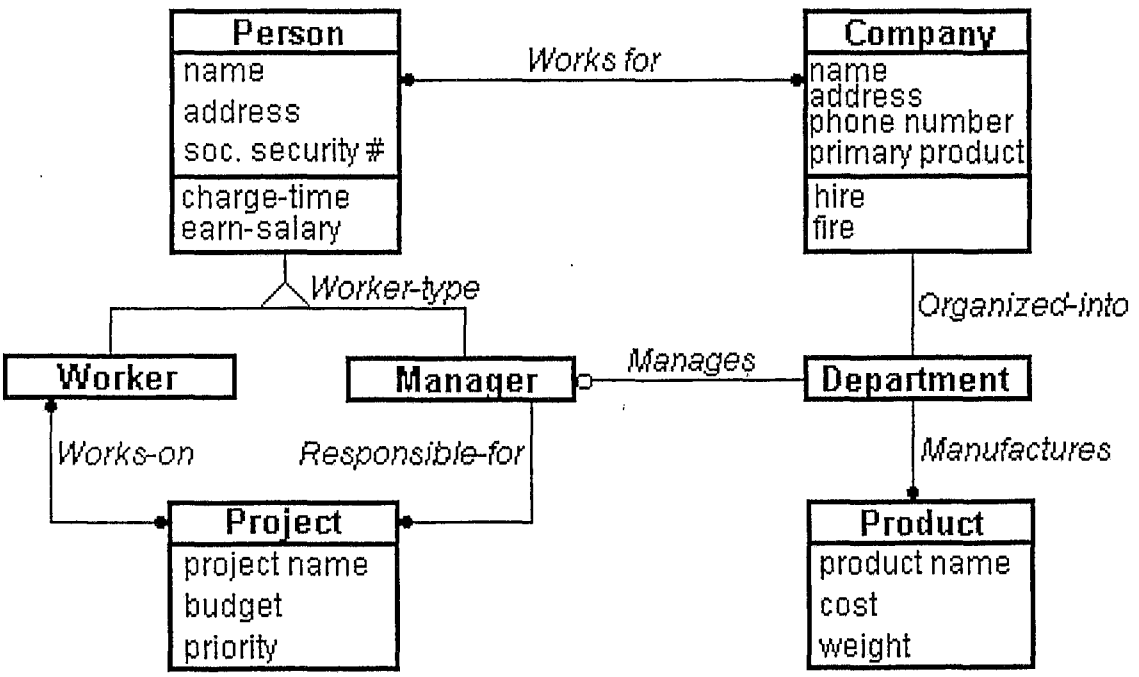

Figure 1. Simple OMT Object Model (after Rumbaugh et al., 1993, Fig. 12.2, p. 272)

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

operations. Given our example, the object PERSON contains attributes of NAME, ADDRESS, and SOCIAL SECURITY NO. and operations called CHANGE-TIME and EARN-SALARY. As can be seen in Figure 1, the OMT methodology displays an object's attributes above its operations.

The relationships between objects are expressed as lines connecting objects and are termed *associations*. In our example, the association between the object class DEPARTMENT and the object class PRODUCT expresses the relationship of MANUFACTURing. More explicitly, OMT includes notation that is capable of indicating the cardinality of such relationships. In the aforementioned example, the single line on the DEPARTMENT end and the solid circle on the PRODUCT end of the MANUFACTUREs association indicates that one department may manufacture many products. Similarly, many WORKERS may WORK-ON multiple PROJECTS. A hollow circle at the end of an association denotes optional cardinality; that is to say, in the company described in Figure 1, a DEPARTMENT may or may not have a MANAGER (in the case of self-managed departments) but a MANAGER is always assigned to manage a single DEPARTMENT.

An inherent component of any object model, and indeed, a primary reason for the robustness of the approach, is the expression of class hierarchies between objects via the type of association used to represent a relationship. One type of hierarchical association, termed a *generalization*, can be seen in the relationship WORKER-TYPE. Classification schemes typically express generalization via the class/member relationship. In Boolean terms, a generalization may be though of an "or-relationship." Generalizations, such as WORKER-TYPE, are designated by links that display a branching character. In this example, the object classes WORKER and MANAGER are subclasses of the more general, superclass PERSON. Because WORKER and MANAGER are subclasses of PERSON, they *inherit* both the attributes (e.g., NAME, PERSON, SOCIAL SECURITY NO.) and operations (e.g., CHANGE-TIME, EARN-SALARY) of the superclass. The generalized nature of the WORKER-TYPE association is the reason that the objects WORKER and MANAGER appear without data or methods in Figure 1. One of the challenges in object-oriented design is balancing the potential trade-offs between the desire to reduce the number of objects used in the modelling of a system while at the same time increasing the opportunities for inheritance of data and methods between object classes (Khoshafian & Copeland 1986, Lieberman 1986, Meyer 1986).

Aggregation, another type of hierarchical association, can be found in the model described in Figure 2.

Classification schemes frequently use aggregation via the expression of whole/part relationships; or, in Boolean terms, an aggregation may be thought of as an "and-relationship." According to the OMT methodology, the aggregation symbol, a diamond appearing on the LAMP end of the association that could be called IS COMPOSED OF, indicates that the object class LAMP is composed of exactly one BASE and one COVER and one SWITCH and WIRING. It should be noted that according to the cardinality of associations expressed in Figure 2, the OMT problem space defined here could not, for example, be used to describe lamps that had two SWITCHes.

Another important distinction between generalizations and aggregations relates to the difference associated with the run-time characteristics of object-oriented systems, or how the hierarchy of data and procedural structures are effectuated by the operating system. Strictly speaking,

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

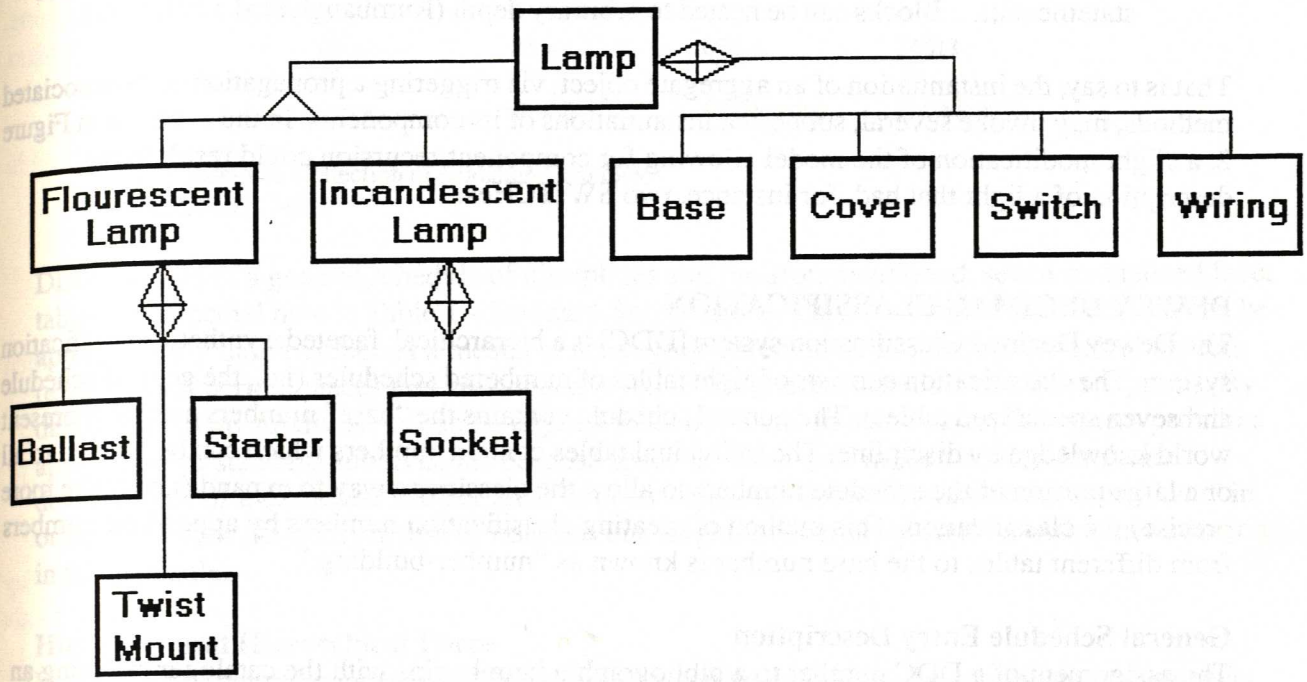

Fig. 2. Aggregation and Generalization (Rumbaugh et al., 1991, Fig.4.2, p. 59)

generalizations refer to abstract classes of objects that *may* occur in the problem space. An information system developed under an object-oriented methodology possesses the necessary structure to logically support objects and their characteristics as identified in the system analysis; however, physical instantiations of these objects are created by the object-oriented system only as they are needed. It is in this way that

> [a]ggregation relates instances. Two distinct objects are involved; one of them is part of the other. Generalization relates classes and is a way of structuring the description of a *single* object. (Rumbaugh 1991, p. 58).

It is only after an instance of a particular object is invoked by the operating system that potential mechanisms for inheritance of attributes and methods becomes activated (Wilkes 1987). To be exceedingly clear on this distinction — the objects in an object model are high level abstractions that describe what *might* be found to exist in the mini-world under study; it is only as certain instances of those abstract model objects are actually encountered that particular objects in the model become *real*. This propagation of methods through the components of an aggregate object is exactly of the type that will be found in our dynamic model.

The final fundamental component of object modelling necessary to understand our proposed model deals with the notion of communicating, aggregate recursive objects (Lamersdorf 1986). In a fashion similar to the notion of recursive procedure calls in a functional programming environment,

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

> [a] recursive aggregate contains, directly or indirectly, an instance of the same kind of aggregate; the number of potential levels is unlimited. ...[For example, a] computer program is an aggregation of blocks, with optionally recursive compound statements;.... Blocks can be nested to arbitrary depth (Rumbaugh et al 1991, p.59).

That is to say, the instantiation of an aggregate object, via triggering a propagation of its associated methods, may invoke several, subnested instantiations of its components. In the example in Figure 2, a slight modification of the model allowing for component recursion could result in the description of a light that had, for instance, two SWITCHes.

DEWEY DECIMAL CLASSIFICATION

The Dewey Decimal Classification system [DDC] is a hierarchical, faceted, synthetic classification system. The classification consists of eight tables of numbered schedules (i.e., the general schedule and seven specialized tables). The general schedule contains the "base" numbers used to represent world knowledge by discipline. The individual tables contain numbers which can be applied to all or a large portion of the schedule numbers to allow the classifier a way to expand (i.e., make more precise) the classification. This method of creating classification numbers by appending numbers from different tables to the base number is known as "number-building."

General Schedule Entry Description

The assignment of a DDC number to a bibliographic item begins with the cataloger selecting an appropriate entry point in the General Schedules. A typical entry point to the DDC rule base might look as follows:

> 287.44-.49 Primitive Methodist Churches, treatment by continent, country locality
> Add to base number 287.4 notation 4-9 from Table 2
> e.g., Wesleyan Methodist Church in Wales - 287.4429

The numeric component of the general schedule entry is termed the *notation* (e.g., "287.44-.49" in the example above), and will eventually be incorporated, either in its entirety or some portion thereof, into the completed DDC number. The *heading* (e.g., "Primitive Methodist Churches..." above) is a caption used to describe the subject the notation represents. *Instructions* are phrases containing action words associated with the number-building process (e.g.,"Add to base number..."). Finally, the *description* (e.g., "Wesleyan Methodist Church in Wales - 287.4429") is additional verbiage associated with an entry that assists a cataloger in the selection of an appropriate base number. All schedule entries contain notation and heading information; the inclusion of instructions and/or descriptions is variable. It should be noted that the instruction set of a chosen general schedule entry *may* **refer** the cataloger to the instructions set of a related entry during the number-building process, thus providing for a "chaining" of instruction sets from one related entry to another.

Facets and Synthetic Classification

Synthetic classification systems are those in which the classification number is constructed from clearly defined and mutually exclusive aspects (facets) of a specific subject. Facets can include discipline, place, time, material, activity. A synthetic scheme consists of a set of tables listing

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

individual facets and a set of rules which prescribes the combining of facets to create complex classification numbers. A synthetic classification number can be decomposed to identify its individual facets, for example:

738 +	09 +	81 +	074 +	74811
↑	↑	↑	↑	↑
Ceramics	Place	Brazil	Collections	Philadelphia

A Philadelphia Collection of Brazilian Ceramics
738.098107474811

DDC consists of a general schedule of disciplines and the aforementioned, seven specialized facet tables[1]. Of special note is Table 1 - Standard Subdivisions. This is a table of numbers that can be applied to any base number as a means of conveying different form, method, or topical aspects (e.g., format, period, place) that are not conveyed in the general schedule entry descriptions. By default, standard subdivisions can be applied to all numbers. In some cases, the system prohibits application of standard subdivisions because space is "reserved" in the notation for future development of a particular subject area. In other cases, the default condition is "overridden" in order to prevent the construction of notation that might inadvertently conflict with other notation in the schedule.

Hierarchy and Hierarchical Force

Hierarchy is a central concept of DDC. Hierarchy can be defined as the subdivision of concepts into more specific (narrower) concepts. Generally, notations at any given level in the hierarchy are a subclass of the numbers whose notation is one digit shorter:

612	Human physiology		
612.1		Blood & circulation	
612.11			Blood
612.111				Red corpuscles
612.112				White corpuscles

Unfortunately, this general pattern does not always hold true. To help prevent unnecessarily long numbers, some areas violate notational hierarchy:

439.5	Scandinavian languages
439.6	West Scandinavian languages - Icelandic
439.7	Swedish
439.8	Danish & Norwegian
439.9	East Germanic languages

In the schedules, DDC uses a formatting device called a "centered heading" to represent the hierarchic relation between terms which are not apparently related hierarchically by the notation.

1. Table 1: Standard subdivisions; Table 2: Geographic areas, historical periods, persons; Table 3: Subdivisions for individual literatures, for specific literary forms; Table 4: Subdivisions of individual languages; Table 5: Racial, ethnic national groups; Table 6: Language; Table 7: Groups of persons

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

DDC exhibits hierarchical force: all of the attributes of one class will apply to all the subdivisions of that class. This is especially important in regards to number-building as number-building behavior of a specific class will hold true for all of its subordinate classes. DDC users are usually trained to trace up the hierarchy of a particular class to determine: 1) if there are any instructions which may be inherited; and 2) that the intended subject fits within the scope of the broader class.

Number-Building

The set of rules used to combine facets to create complex classification numbers are imbedded in the classification as "Add instructions." For example:

398.9 Proverbs
Add to base number 398.9 notation 1-9 from Table 6, e.g., French proverbs 398.941.

This allows the subdivision by language (*Table 6*) of works of proverbs. In the example, the French language facet (notation "41" in *Table 6*) is applied to the base notation for Proverbs. Note that this instruction gives the source for appended notation (*Table 6*) and the range within that table (1-9). Appended numbers can be taken from tables or from the general schedule. If there is no explicit source statement, then the main schedule is the source.

Number-building can be an iterative process as the notations referred to in an add instruction may also contain add instruction referring to other notations. Additionally, add instructions can contain multiple steps thereby determining a facet citation order for that class:

324 Political science (Politics & government)
.2941-.2948 Parties in states and territories of Australia
Add to base number 324.294 the numbers following -94 in notation 941-948 from Table 2 for state or territory, e.g., parties in New South Wales 324.2944; then to the result add the numbers following 324.294 in 324.294001-324.29409, e.g., Labour Party in New South Wales 324.294407.

In this case, the first subdivision is for the political division and the second for the political party. In the first instruction, note that in addition to providing the source (*Table 2*) and range (941-948), the user is also instructed to ignore the first two digits (94) in the notation found. These three pieces of information: source, range and ignored notation are critical to the number-building process.

Additionally, some classes contain an internal table composed of subdivisions specific to that class and its subdivisions:

333 Economics
.7 Natural resources & energy
Except for additions, changes, deletions, exceptions shown under specific entries, add to notation for each term identified by * as follows:
1 General topics
11 Reserves (Stock, Supply)
12 Requirements (Need, Demand)
[etc.]

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

This instruction specifies that those subdivisions of 333.7 marked with "*" can additionally have an internal table subdivision applied.

The augmented number which is the result of following add instructions at the base number can be further specified by applying a standard subdivision. Unlike other synthetic schemes, DDC allows only one standard subdivision to be appended to an augmented number. If there is still more than one aspect which can be applied to the classification, the user consults a precedence table to determine which aspect to classify. The user appends the standard subdivision to the augmented number and then follows the corresponding add instructions to complete the number.

As listed in *Table 1*, all standard subdivisions begin with a single zero. This zero serves as a standard subdivision facet indicator. However, in some areas of the main schedule facets other than those in standard subdivisions are introduced by a single zero. For example:

271		Religious congregations and order in church history Use 271.001-271.009 for standard subdivisions
271	.01-.09	Specific kinds
	.01	Contemplative
	.02	Eremitical
	.03	Teaching
	[etc.]	

The "use note" is instructing the user that for this class, the facet indicator for a standard subdivision is two zeros, so that the standard subdivisions will not be confused with kinds of religious congregations. Note that this particular attribute of a class is not inherited to the subdivisions (i.e., the standard subdivision of 271.1, "Benedictines," is introduced by a single zero, not two zeros).

In DDC, the classification process can be described in five steps:

1) Select a base number;
2) Augment the base number by following the add instructions specific to that base number;
3) Determine if there are additional facets that can be applied by use of a standard subdivision;
4) Select a standard subdivision;
5) Further specify the augmented number by following add instructions specific to that standard subdivision.

The object model discussed in the next section of this paper has the immediate potential to assist the user in the third and fifth steps of this process; that is, the mechanical process of synthesizing a classification number. With the incorporation of other information about the classification (e.g., *description* information), this data structure has the potential to assist users in the entire process of creating a classification.

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

OBJECT MODEL & NUMBER-BUILDING PROCESS

Static Object Model

Underlying assumptions affect the model design. Thus, it should be made clear that references are made specifically to the twentieth edition of DDC (Comaromi 1989), that only a subset of the entire classification (specifically, the General Schedule, Table 1 and Table 2) is represented by the object model presented here, and that there is not an exact match between the structure of a DDC classification entry and the model's structure of the same information (i.e., standard subdivision information isn't represented at each DDC classification entry, but is instead represented through summary notes and footnotes). The number building operations presented here are a subset of all the potentially possible numbering building processes. Additionally, the model does not attempt to simulate or explain a classifier's decision making process but instead models the number combining properties of a synthetic classification scheme, such as DDC.

As can be seen in Figure 3, the model is predicated upon an external cataloging mini-world in which a bibliographic cataloging record is being produced. According to OMT descriptive terminology, that record is an aggregation of such elements (fields) as author, title, physical description and subject content of the book. The cataloger, the catalog record and the cataloger's interaction with the number-building process remain external (i.e. below the dotted line) to our object model. It is the "environment" within which the number-building process happens. The ENTRY objects correspond to entries in the classification schedule. The object WORKING DEWEY #, analogous to a cataloger's worksheet, is an "object in process" until application of the rules has been completed to produce the classification element of the catalog record. Only at the point in time when the cataloger begins interaction with the DDC rule set, is the object model set in motion.

The four generalized subclasses (SCHEDULE ENTRY, TABLE 1 ENTRY, TABLE 2 ENTRY, INTERNAL TABLE ENTRY) represented in the model share a common structure which is generalized to an abstract superclass ENTRY. The main distinction between these object classes are the source of the individual classification instances (i.e. classification number). All the subclasses inherit the attributes of the abstract superclass ENTRY, namely NOTATION, HEADING, INSTRUCTIONS, and DESCRIPTION, to which the operations or methods of ADD, DELETE, or EDIT will be applied. There are minor distinctions associated with two of the generalized subclasses. The subclass object SCHEDULE ENTRY has two additional attributes, STANDARD SUB and # OF ZEROS, which contain information about the standard subdivision appending behavior of base/augmented numbers. The subclass object INTERNAL TABLE ENTRY has the further unique attribute RANGE REFERENT, which refers the cataloger to one of a set of numerically disjoint sets of notation that may reside inside a SCHEDULE or TABLE entry. INTERNAL TABLES apply solely to the entry (and its descendants) where they reside.

In addition, each of the object subclasses exhibits a recursive generation, labelled as BROADER_THAN (i.e., "BT") on Figure 3, which represent the hierarchical relation between individual classification instance. Therefore, a single object class (e.g., SCHEDULE ENTRY) can be used to define the structure and relationships of an entire table of hierarchical classifications:

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

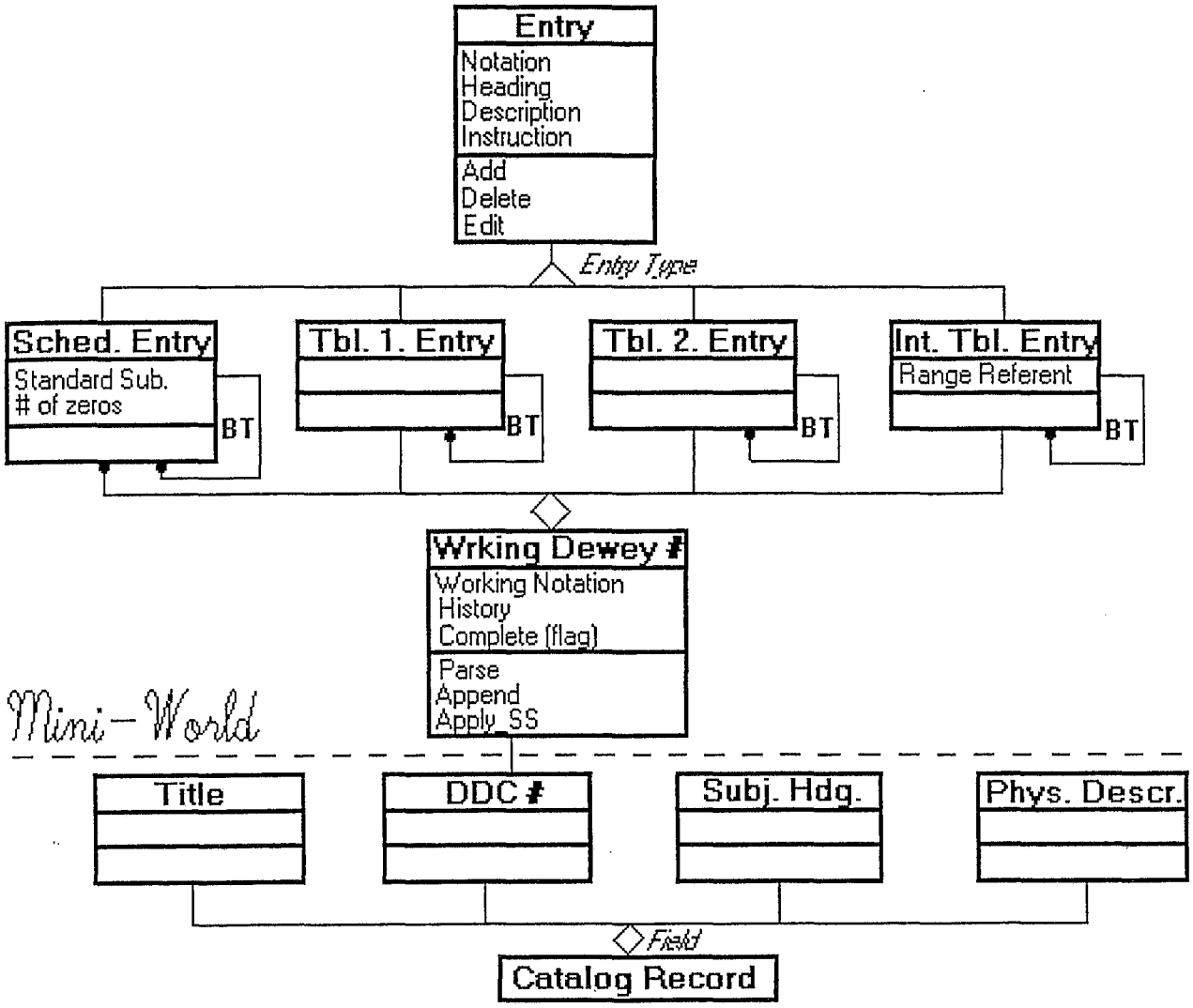

Figure 3. Dewey Object Model

Dynamic Object Model

The object model attempts to simulate the building of classification numbers by representing DDC schedule entries as instances of particular object subclasses (e.g., SCHEDULE ENTRY). The entry will contain the attributes of the generalized object superclass ENTRY (NOTATION, HEADING, DESCRIPTION, INSTRUCTIONS) and any attributes specific to the object subclass. The hierarchical arrangement of ENTRIES in DDC permit a particular Dewey classification number to inherit any attribute values of its broader classification as can be seen in the following instance diagram [See Figure 4]. Note that the standard subdivision attribute, STD SUB (is standard subdivision allowable) is left blank in the schedule entries for notations 623, 624 and 625. The actual value of this attribute for these instances is Y, the value inherited (by default) from the

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

schedule entry with notation 62. Also note that the value of # OF ZEROS (number of zeros used to serve as a standard subdivision facet indicator) is explicitly given as 1 for these notations; the value is not inherited from its broader classification. One of the advantages of using object modeling to represent hierarchical classification is that the characteristic of inheritance which is natural to hierarchic classification schemes can be easily represented in an object model through the use of recursive generalizations. An inherited characteristic is only represented once in the model at the highest, most general level of classification to which the characteristic applies.

Based on examination of the book's contents, using such determinants as the title, table of contents, and indexes, the cataloger determines an initial instance of the SCHEDULE ENTRY object, that is, the "base number" which serves as an entry point into DDC, representing the broad subject area content of the book. The completed number will be built. To represent the number being built through its various stages, there is an aggregation object, WORKING DEWEY #, which has three attributes and corresponds to a cataloger's worksheet. WORKING NOTATION represents the notation built up to a given point in the process, HISTORY documents the steps which brought it to that point and COMPLETE is a flag to signal there are no further rules which must or may be applied to extend the number-building process. Note that WORKING DEWEY # contains no values until the classification process begins with the selection of a base notation.

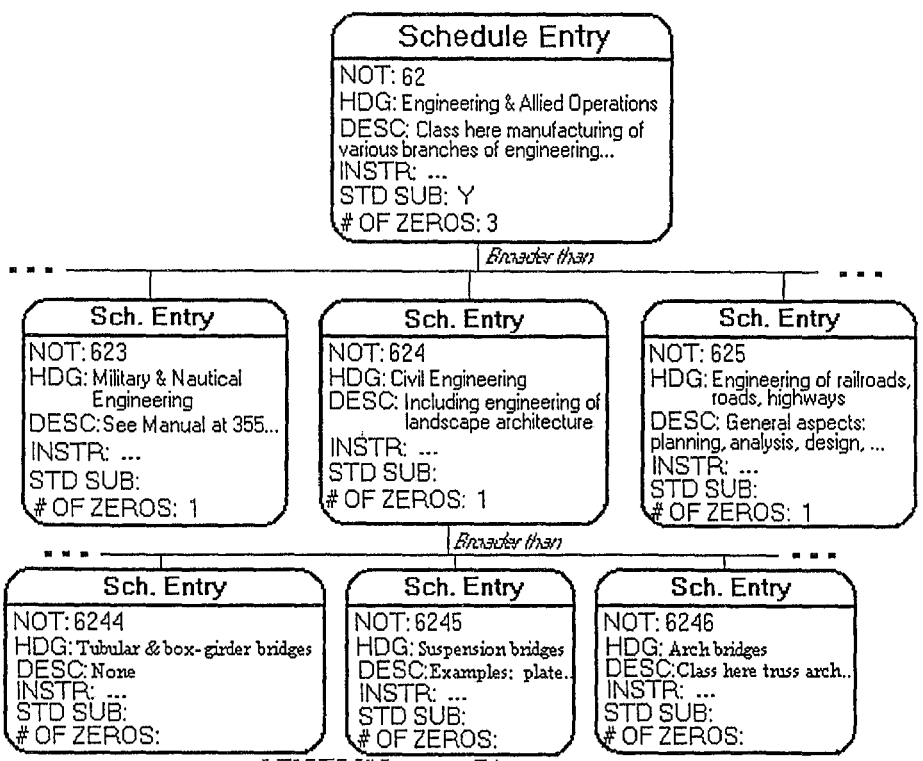

Figure 4. SCHEDULE ENTRY Instance Diagram
(Ellipses indicate omission for brevity; blanks indicate inheritance)

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

Using the operation PARSE, the WORKING DEWEY # will read and interpret the NOTATION to determine which method should be applied and what instances of the DDC rule base must be examined next. Parsing the rule base will direct the WORKING DEWEY # to the appropriate instructions by textual clues and enable actualization of the instruction. The APPEND algorithm produces and annexes the incremental segment of the number found at that point in the rule base to that held on the "worksheet" and provide the parameters within which the rule should be applied. In some cases, the rule will cite a range of possible numbers which might be applied to the WORKING DEWEY # at a given point in the process and refer to another section of the rule base to be applied. In the case of multiple-step add instructions, it may be necessary for the WORKING DEWEY # to hold a number segment or instruction in reserve until it has performed another set of PARSE and APPEND operations in accordance with the rules and, subsequently, to return for processing the rule for that segment.

After the construction of the augmented base number, the APPLY_SS operation is performed to determine the applicability of a standard subdivision (i.e., Is a standard subdivision allowable given a specific base notation?) and to process the appending of an appropriate standard subdivision. At this point the COMPLETION flag is triggered signaling the cataloger that the full classification number is ready for use. The cataloger, although external to this model, does interact with both the APPEND and APPLY_SS operations at appropriate decision points to enable continuance of the number-building process.

CONCLUSION

There are two characteristics of object-oriented design which especially lend themselves to representation of hierarchic, synthetic classification schemes. First, the hierarchy implicit in the recursive Broader-Than relation allows for a more natural representation of hierarchy than is possible through the use of linking fields in a relational database structure. It also allows for representation of hierarchy not based on "notational hierarchy," especially around centered headings where notational hierarchy is often violated. Secondly, it allows for the "behavior" of an individual classification to be closely associated with its corresponding notation. In traditional relational database design, the procedures which act on the database are separate from the data. Due to the fact that very specific algorithmic procedures are associated with individual data instances, any programs developed in association with a traditional relational structure would quickly become lists of "exceptions" to the general number-building process. Additionally, in strongly faceted systems, entries can be grouped together by facet through the use of the "entry object" (e.g., Table 1). Behaviors specific to that facet need only be specified in the object subclass definition or in its most general entries to be inherited by all of its subclassifications.

The model presented here demonstrates the typical robustness of an object-oriented design. Given only the text of the DDC classification schedule, two distinct classes of objects (i.e., ENTRIES and WORKING DEWEY #), and some simple, algorithmic procedures (i.e., PARSE, APPEND, APPLY_SS), our model is able to accurately describe the dynamics of the DDC number-building process.

Alternatively, the object model could be modified so that the DDC "Add instruction" would be represented as a method associated with a specific classification instead of as an attribute which is

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

manipulated by WORKING DEWEY #. WORKING DEWEY # would continue its role as "historian" and "supervisor" of the number building process, but would no longer be responsible for the execution of every add instruction. However, the prerequisite for this model is that all add instructions are translated from text into a "method" form (PARSE) prior to the development of the database. In the first model, the parsing of the add instruction text happens when the classification instance becomes active. In the second model, all add instruction text is parsed before database implementation and the resulting parses are the methods associated with the individual classification instances. In both cases, the add instructions are still uniquely associated with the particular object instances.

The most straightforward application of such a model would be in the development of a decision support system [DSS] to be used by entry level cataloguers. Indeed, the ability of the model to allow for the stepwise construction and deconstruction of DDC notation-in-process provides for exactly the sort of alternative-generating capabilities that are integral to the operation of such DSSes (Brill 1990). Some processing will be required to map the DDC classification schedule data into the object model, but other applications, such as information retrieval and classification mapping, have the potential to exploit these object-oriented data structures.

REFERENCES

Auty, David. *Object Oriented Programming Systems (OOPS) and Frame Representations, an Investigation of Programming Paradigms Final Report.* Houston, TX: Research Institute for Computing and Information Systems, July 31, 1988. NASA, NASA-CR-186084.

Bhalla, Neelam. "Object-oriented data models: a perspective and comparative review," *Journal of Information Science* 17(3) (1991): 145-160.

Blaha, Michael R., William J. Premerlani and James E. Rumbaugh. "Relational Database Design Using an Object-Oriented Methodology," *Communications of the ACM* 31(4) (April 1988): 414-427.

Bonar, Jeffrey, Robert Cunningham and Jamie Schultz. "An Object-Oriented Architecture for Intelligent Tutoring Systems." *OOPSLA'86: Sigplan Notices Special Issue* (November 1986), 269-276.

Booch, Grady. "Object-Oriented Development," *IEEE Transactions on Software Engineering* SE-12(2) (February 1986): 211-221.

Brill Jr., E. Downey, et al. "MGA: A Decision Support System for Complex, Incompletely Defined Problems," *IEEE Transactions on Systems, Man, and Cybernetics* 20(4) (July/August 1990): 1134-1141.

Comaromi, John P., ed. *Dewey Decimal Classification and Relative Index - 20th ed. Vols. 1-4,* by Melvil Dewey. Albany, NY: Forest Press, 1989.

Ewing, Juanita J. "An Object-Oriented Operating System Interface." *OOPSLA'86: Sigplan Notices Special Issue* (November 1986), 45-56.

Frank, Andrew U. "Multiple Inheritance and Genericity for the Integration of a Database Management System in an Object-Oriented Approach," in Dittrich, K.R., ed. *Lecture Notes in Computer Science.* Vol. 334, *Advances in Object-Oriented Database Systems,* Berlin: Springer-Verlag, 1987, 268-273.

Guzev Y.S. and V.A. Titov. "Object oriented approach to creating intellectual information systems," *Nauchno Tekhnicheskaya Informatsiya Seria Dva* 11 (1991): 19-26.

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

Hinsch, Kathryn et al. "Object-oriented programming: its role in computing," *Library Software Review* 9(1) (January/February 1990): 18-24.

Izygon, Michel. *Advanced Software Development Workstation - OOPSLA '92 Conference Trip Report.* Houston, TX: Research Institute for Computing and Information Systems, December 31, 1992, NASA, NASA-CR-192811.

Kochtanek, Thomas R. "Procedural logic versus object-oriented logic in library automation instruction," *Journal of Education for Library and Information Science* 28(1) (Summer 1987): 55-57.

Khoshafian, Setrag N. and George P. Copeland. "Object Identity," *OOPSLA'86: Sigplan Notices Special Issue* (November 1986), 406-416.

Lamersdorf, W. "Communicating Recursive Objects," in *Proceedings of the 1986 International Workshop on Object-Oriented Database Systems Held in Pacific Grove California 23-26 September 1986*, Washington D.C.: IEEE Computer Society Press, 1986, 225-226.

Lieberman, Henry. "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems." *OOPSLA'86: Sigplan Notices Special Issue* (November 1986), 214-233.

Liu, Songqiao. "The Automatic Decomposition of DDC Synthesized Numbers," Ph.D. diss., University of California, 1993.

Meyer, Bertand. "Genericity Versus Inheritance." *OOPSLA'86: Sigplan Notices Special Issue* (November 1986), 391-405.

Parsaye, Kamran et al. *Intelligent Databases: Object-Oriented, Deductive Hypermedia Technologies.* New York: John Wiley & Sons, Inc. 1989.

Popken, Douglas A. *Hierarchical Modeling and Process Aggregation in Object-Oriented Simulation* Brooks Air Force Base, Texas: Air Force Systems Command - Armstrong Laboratory, May 1991. AL-TP-1991-0020.

Rostek, Lothar and Dietrich H. Fischer, "Modelling a thesaurus in a frame system with graphical interface," *Nachrichten fur Dokumentation* 39(4) (August 1988): 217-226.

Rumbaugh, J. E. et al. *Object-Oriented Modelling and Design.* Englewood Cliffs, NJ: Prentice-Hall, 1991.

Schmitz, Esser W. "New approaches to thesaurus application," *International Classification* 18(3) (1991): 143-147.

Straube, David D. and Ozsu M. Tamer. "Queries and query processing in object-oriented database systems," *ACM Transactions on Information Systems* 8(4) (October 1990): 387-430.

Wilkes, Wolfgang. "Instance Inheritance Mechanisms for Object-Oriented Databases," in Dittrich, K.R., ed. *Lecture Notes in Computer Science.* Vol. 334, *Advances in Object-Oriented Database Systems,* Berlin: Springer-Verlag, 1987, 274-279.

ISSN: 2324-9773

Giguere, M. (1994). Object-Oriented Representation ofDDC and the Number-Building Process. 5th ASIS SIG/CR Classification Research Workshop, 79-94. doi:10.7152/acro.v5i1.13778

ISSN: 2324-9773