Department of Education, Federal Student Aid

Data Model Standards and Guidelines, Registration Policies and Procedures

Version: 1.1

Revision

September 2007

START HERE GO FURTHER FEDERAL STUDENT AID

Table of Contents

1.0 Overview 1
1.1 Introduction 1
1.2 Scope 1
1.3 EDM Data Modeling Environment 1
1.4 Benefits 2
2.0 Modeling Standards and Guidelines 3
2.1 High Level Data Model Definitions 3
2.1.1 Entity Data Model 3
2.1.2 Entities 4
2.1.3 Relationships 4
2.1.4 Attributes 4
2.1.5 Class Word 4
2.1.6 Meta Data 5
2.2 Data Standardization 5
2.3 Common Standards and Guidelines 6
2.3.1 Naming Standards 6
2.3.2 Definition Standards 6
2.4 Unique Data Object Naming and Definition Standards and Guidelines 7
2.4.1 Entity versus Object Modeling 7
2.4.2 Data Object Naming Conventions 8
2.4.3 Data Object Definitions 11
2.4.4 Generic Class Words 11
2.5 Conceptual Data Modeling (CDM) Standards and Guidelines 12
2.5.1 Conceptual data model packaging: 13
2.5.2 Conceptual data model level of detail: 13
2.5.3 CDM Review Template 14
2.6 Logical Data Modeling (LDM) Standards and Guidelines 14
2.6.1 LDM Review Template 15
2.7 Physical Data Model (PDM) Standards and Guidelines 15
2.7.1 PDM Review Template 17
3.0 Data Model Registration (DMR) 18
3.1 DMR Policy 18
3.2 DMR Procedures 18
3.2.1 DMR Procedures for existing applications 18
3.2.2 DMR Procedures for future developments in support of TSV 19

3.2.3 DMR Process Flows 20
3.3 Roles and Responsibilities for the DMR Process 21
3.3.1 DMR Responsibilities of the EDM Team 21
3.3.2 DMR Responsibilities of the System Developer 22
3.4 Registration Requirements for Data-Related Artifacts 22
3.4.1 Data Model Meta Data Registration Requirements 23
3.4.2 Data Model Entity or Table Meta Data Registration Requirements 23
3.4.3 Attribute/ Column Meta Data Registration Requirements 23
3.4.4 Registering COTS Meta Data Requirements 23
3.4.5 Registering all the supporting documents related to data models 24
Appendix A. Glossary 25
Appendix B. Abbreviations / Acronyms 26
Appendix C. References 27
Appendix D. Generic Elements/ Class Words 28
Appendix E. Entity or Table Metadata Requirements 32
Appendix F. Attribute or Column Metadata Requirements 34
Appendix G. Conceptual Data Model Review Template 36
Appendix H. Logical Data Model Review Template 37
Appendix I. Physical Data Model Review Template 42
Appendix J. Information Engineering (IE) Notation 44
Appendix K. ER/Studio Enterprise Data Dictionary (EDD) 46
Appendix L. ER/Studio Repository 47
Appendix M. Converting Data Types 57
Appendix N. XML Crosswalk 58
Appendix O. Data model registration process and ECDM Change Request process flows 59
Appendix P. Data model registration templates 64
Appendix Q. Data Modeling Essentials 69
Appendix R. Defining the Data - Constructing a well-written data element definition 77

List of Figures

Figure 1: Registration process flow diagram. 20
Figure J-1: IE notation. 45
Figure L-1: ER/Studio Repository menu. 47
Figure L-2: Creating folder structures under the projects. 48
Figure L-3: Managing repository projects. 49
Figure L-4: The Wizard to create project names. 50
Figure L-5: Close-up of the Wizard's dialog box. 50
Figure L-6: Entering the project name and description. 51
Figure L-7: Close-up of entering the project name and description. 51
Figure L-8: The Repository Project Center. 52
Figure L-9: Close-up of the Repository Project Center. 52
Figure L-10: Creating a project repository. 53
Figure L-11: Close-up of creating a project repository. 53
Figure L-12: Close-up of ER/Studio Repository Diagrams menu. 55
Figure L-13: Adding a diagram to the ER/Studio Repository. 55

List of Tables

Table 1: Data object names matrix. 9

Table 2: Naming standards matrix. 10

Document History

Change Number	Date	Reference	A M D[1]	Title or Brief Description	Author	Change Request Number
1.0	6/29/2007			Final draft		
1.1	9/24/2007			Added process flows for DMR		

[1] A(dd), M(odify), or D(elete)

1.0 Overview

1.1 Introduction

Data are principal Federal Student Aid resources, which like other organizational resources, must be managed effectively. The use of standardized data enhances the interoperability among Federal Student Aid information systems, facilitates increased data sharing, reduces data handling costs and leads to better data accuracy, consistency and timeliness. The policies and procedures of this document provide the framework necessary to maximize data sharing and exchange opportunities and to enable standardized data modeling throughout Federal Student Aid.

The data required by Federal Student Aid is maintained in the Enterprise Data Dictionary (EDD) and is input to the data modeling process. Data models contain data entities, the relationships between the various data entities, and data entity attributes. One output of the data modeling process is the graphical representation of the data model as an Entity Relationship Diagram (ERD), specifically, of the Enterprise Conceptual Data Model (ECDM) and Enterprise Logical Data Model (ELDM).

1.2 Scope

The scope of this document is to:

- Define the data modeling standards and guidelines including naming conventions.
- Identify the policies and procedures for Data Model Registration (DMR) at Federal Student Aid. These policies and procedures address how to capture all the existing Federal Student Aid's business/legacy systems data models and how to develop future data models for new business systems across Federal Student Aid.

This document illustrates a clear, effective process to be utilized by Federal Student Aid for adopting, developing, implementing and maintaining data modeling and registration standards for the purpose of information sharing and exchange. This document, in combination with the "Data Standardization Policies and Procedures" and the "Draft Enterprise Data Management Data Policies", supports effective EDM operations across Federal Student Aid. Throughout this document, standards are preceded by an (S), and guidelines are preceded by a (G).

1.3 EDM Data Modeling Environment

Federal Student Aid is consistent with major federal government efforts to define a common framework or reference model to be used in describing data modeling standards. The Office of Management and Budget (OMB) defined the Federal Enterprise Architecture Data Reference Model (FEA DRM) as a means to provide a common and consistent way of categorizing, and describing data to facilitate data sharing and integration. The concept of capturing and describing data follows the standard developed by the International Standardization Organization (ISO)/International Electrotechnical Commission (IEC).

Federal Student Aid's ECDM and EDD are maintained in Embarcadero's ER/Studio.

ER/Studio is used for the development of the logical data models and the physical database design and construction. It has the following features:

- Strong logical model capabilities
- Bidirectional synchronization of logical and physical models
- Automatic database construction
- Reverse-engineering of databases
- HTML-based documentation and reporting facilities
- Data modeling repository for cooperative team modeling.

1.4 Benefits

The data modeling standards, policies, and procedures outlined in this document establish a sound foundation for enterprise data modeling:

- All the existing business systems / legacy data models are available in one place.
- All the future developments for new business systems will follow the same policies and procedures for data standardization and data model development.
- The EDM Team will review and validate the new business system data models for compliance, thereby increasing the consistency and transparency of data architecture across the enterprise.
- The common EDD is available for use by future business systems, thereby improving data exchange across all the business systems.
- Seamless data integration between all the new business systems is enabled.
- Data modeling standards are specific rules for the development and modification of the names, definitions, and other metadata classes, attributes and data models.
- Data modeling standards establish a required level for the correctness, consistence, and completeness of data models.
- Data modeling standardization focuses on universal understanding of data and their business context throughout the enterprise rather than within the confines of a particular environment. Policies and procedures are instructions and recommendations for the successful implementation of data modeling standards.

Following the data modeling standards will reduce the time required for users and system developers to identify information assets appropriate for decision making, and will permit identification and reduction of redundant data stores, interfaces and their associated costs.

2.0 Modeling Standards and Guidelines

The EDM Team develops and distributes standards and guidelines for conceptual, logical, and physical data modeling including data naming standards. These standards and guidelines address the level of detail required for each particular type of data model. A ***conceptual data model (CDM)*** is a clear and accurate visual representation of the business of an organization. It includes all major entities and relationships and does not contain much detailed level of information and is often used during the planning phase of a project. Section 2.5 of this document contains standards and guidelines for conceptual data model design. A ***logical data model (LDM)*** is more complete and detailed than a CDM and has more applicable data standards and guidelines. Section 2.6 of this document contains standards and guidelines for LDM design. A ***physical data model (PDM)*** reflects implementation considerations such as performance, denormalization, and data access path, and it can be different from the LDM. Section 2.7 of this document contains standards and guidelines for PDM design.

The EDM Team will help with application data model development on an as-needed basis. It will review and validate each conceptual, logical, and physical application data model using the standards-based templates in Appendix G, H, and I respectively. The following standard determines the different delivery formats for data models:

- (S) All emerging or legacy conceptual, logical, and physical application data models will be provided in the format of the data modeling tool used for the modeling effort (ER/Studio is the standard tool). Each model will also be prepared in a Word-document report. This Word report is a generic format that can be shared across the enterprise regardless of the availability and/or accessibility of the modeling tool.

- (S) The data modeling notation (cardinality) for relationship information between entities must follow the ***Information Engineering (IE)*** standards. Appendix J provides additional detailed information about IE notation, such as its graphical presentation and how to interpret the individual symbols.

2.1 High Level Data Model Definitions

The Entity-Relationship Data Model (ERDM) is a data model that views the real world as consisting of entities and relationships. The ERD is the visual representation of these concepts. The basic constructs of the ERDM are entities, relationships, and attributes. Entities are concepts, real or abstract, about which information is collected. Relationships are associations between the entities. Attributes are properties, which describe the entities. For detailed information visit the "Introduction to Data Modeling" provided by the University of Texas[2].

2.1.1 Entity Data Model

The Entity Data Model specifies the conceptual model of the data via the ERDM. The ERDM views the real world as entities and relationships. An ERD is commonly used as a tool to discuss and document business information in a relational format, which is used for database design. The utility of the data model is:

[2] University of Texas; Introduction to Data Modeling: http://www.utexas.edu/its/windows/database/datamodeling/dm/erintro.html

- It is simple and easy to understand with a minimum of training.
- Data modelers and database designers can use the model to communicate the design to the end user.
- It can be used as a design plan by the database developer to implement a data model in specific database management software as the constructs used in the ERDM can easily be transformed into relational tables.

2.1.2 Entities

Entities are the principal data object about which information is to be collected. Entities are usually recognizable concrete, tangible or abstract concepts, such as person, places, things, or events important to the organization's business. They usually refer to objects that are relatively stable and long-lived. Some specific examples of entities are PERSON, ORGANIZATION, or LOAN. An entity is analogous to a table in the relational database model. Properties describe the entity by giving it a name and type (primary or associative entity.) Entities must have:

- A name complying with Federal Student Aid's naming conventions.
- An unambigious succint definition.
- A minimum of two attributes: one unique identifier (key) and one or more descriptive attribute.
- A minimum of one relationship to another entity.

2.1.3 Relationships

A *Relationship* represents an association between two or more entities. An example of a relationship would be: "employees are assigned to projects." Relationships can be either one-to-one, one-to-many, many-to-many, zero-to-one, or zero-to-many. Relationships between entities are named, which defines the purpose of the relationship. For more information about relationships see Appendix J.

2.1.4 Attributes

Attributes describe the entity of which they are associated. A particular instance of an attribute is a *value*. For example, "Jane" is one value of the attribute "First Name". Attributes can be classified as identifiers or descriptors. Identifiers are more commonly called *keys*, which uniquely identify an instance of an entity. A descriptor describes a non-unique characteristic of an entity instance.

Each attribute is the physical representation of the column of a table in the relational database model.

2.1.5 Class Word

A class word is a noun that prescribes a definition for a general category of data. A class word designates the category of data into which a data element fits. Examples of class words are "Code," "Name," and "Quantity." A complete list of approved and recommended class words is available in Section 2.4.4.

2.1.6 Meta Data

Meta data involves the capture and presentation of the meaning and context behind the data. Meta data is everything about the data that turns it into information that is useful to the business. Data elements have definitive characteristics that quantify, identify, or describe a concept. For example, data elements have names, definitions, and valid values. Unit of measure, e.g., feet, tons, miles per hour, etc., is a characteristic of a data element, and as such is an item of meta data. Meta data are data about data. Meta data are data or facts about data elements.

2.2 Data Standardization

Federal Student Aid manages data in two different environments:

- The XML Registry and Repository and
- The Data Model Repository for relational data models (tabular data[3])

Federal Student Aid decided to keep the data consistent between the two environments. To enable this synchronization Federal Student Aid

- Correlated the components of the XML Registry and Repository with the components of a data model (see Appendix N - XML Crosswalk).
- Mapped XML data types to data modeling data types (see Appendix M - Converting Data Types).

Data Standardization defines consistent requirements for meta data and applies them to common data structures through a process of review and validations. With Data Standardization being a combination of how data is described, categorized and shared, the following three major areas reflect this as described in the FEA DRM:

> [4]***Data Description:*** Provides a means to uniformly describe data, there by supporting its discovery and sharing.
>
> ***Data Context:*** Facilitates discovery of data by grouping it according to defined taxonomies. Also, enables the definition of authoritative data assets with in a Community of Interest (COI.)
>
> ***Data Sharing:*** Supports the access and exchange of data where access consists of ad-hoc requests (such as a query of a data asset), and exchange consists of fixed, re-occurring transactions between parties. It is enabled by capabilities provided by both the Data Context and Data Description standardization areas.

[3] Tabular Data: is "data in a table", organized in rows and columns (table structure).
[4] Federal Enterprise Architecture – Data Reference Model, November 2005

2.3 Common Standards and Guidelines

Data model entities, tables, attributes, and columns must be described at a sufficient level of detail to ensure that they are discrete and clearly understood. To do this, each attribute or column name must end with an approved or recommended generic element/class word as discussed in Section 2.4.2. In addition, each entity and each attribute or column has a field layout format consisting of mandatory, recommended, or optional metadata requirements as explained in Appendices E and F.

2.3.1 Naming Standards

Ten common naming standards are equally applicable to the data domains. The order of the standards listed below does not imply any order of importance. In summary, a name must:

1. Be correct, that is, both functionally and technically accurate.
2. Be clear, avoiding the use of vague terms such as "handle" or "process."
3. Be concise, using the fewest number of words possible, avoiding articles and needless prepositions.
4. Be unique, avoiding wording similar to that of any other name.
5. Be atomic, representing only a single concept.
6. Contain only letters of the alphabet, numbers, and word separators.
7. Follow the specified format for names as presented below in Section 2.4.
8. Reflect common terminology used throughout Federal Student Aid.
9. Use complete names wherever possible instead of abbreviations or acronyms.
10. Use only approved abbreviations or acronyms when the data modeling tool restricts the length of the name.

2.3.2 Definition Standards

Any name used in the data domains requires a corresponding definition. Because the definition is the sole justification for the existence of the name, the definition must be developed at the same time that the name is created.

As with the above naming standards, and regardless of the particular domain, the following 7 definition standards apply throughout the enterprise. Once again, the sequence of the standards listed below does not imply any order of importance. In summary, a definition must:

1. A definition should be unique and distinguishable from every other data element definition.
2. A data definition should be precise, concise, and unambiguous. The definition should be clear enough to allow only one possible interpretation.
3. The data definition should be expressed in the singular.
4. The definition should include the essential meaning or primary characteristics of the concept.
5. The definition should only use abbreviations when necessary and the abbreviation must be commonly understood.

6. The definition should not contain any embedded definitions or underlying concepts of other data elements/terms/concepts.
7. The definition should not include statements about why and how a data element is used.

Additional information on how to create a well-written definition go to Appendix R of this document.

2.4 Unique Data Object Naming and Definition Standards and Guidelines

This subsection and all of sections 2.5, and 2.6 refer to the naming and definition standards for data models and data objects as well as to entity modeling.

2.4.1 Entity versus Object Modeling

Entity modeling: is a technique used to describe data in terms of entities, attributes, and relationships as described before in Section 2.1.1. Entities are classes of persons, places, or things important to the enterprise business. The entity model provides the most succinct, abstract, and permanent description of the enterprise. Entity modeling occurs at the conceptual level in the conceptual entity model, and at the logical level in the logical entity model.

For example: The ER model views the real world as a construct of entities and association between entities.

Entities are the principal data object about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract, such as person, places, things, or events, which have relevance to the database. Some specific examples of entities are PERSON, ORGANIZATION, and INVOICE. An entity is analogous to a table in the relational model. An *independent entity* is one that does not rely on another for identification. A *dependent entity* is one that relies on another for identification

An *entity occurrence* (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table.

A ***Relationship*** represents an association between two or more entities.

An example of a relationship would be:

- Employees are assigned to projects
- Projects have subtasks
- Departments manage one or more projects.

Relationships are classified in terms of degree, connectivity, cardinality, and existence.

Object Modeling: is represented graphically with object diagrams that contain object classes. Classes allow the characteristics of objects to be abstracted and applied to similar cases. A class is a description of the attributes, operations, and relationships that define a set of objects based on the class.

In object technology, an entity is one type of analysis class. An entity class is persistent, having attributes and relationships. Real-life entity objects are instances of entity classes. When analysis is completed and design begins, an entity class becomes an object class. Because of all their similarities, the terms "entity" and "class" are frequently used interchangeably. This is correct for naming and definition standards and guidelines, but there are some other minor technical differences.

For example: A ***Class diagram*** gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static -- they display what interacts but not what happens when they do interact. UML class notation is a rectangle divided into three parts: class name, attributes, and operations. Relationships between classes are the connecting links.

In general class diagram has three kinds of relationships.

- **Association** -- a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.
- **Aggregation** -- an association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole.
- **Generalization** -- an inheritance link indicating one class is a superclass of the other. A generalization has a triangle pointing to the superclass.

An association has two ends. An end may have a **role name** to clarify the nature of the association. For example, an **OrderDetail** is a line item of each **Order**.

A ***navigability arrow*** on an association shows which direction the association can be traversed or queried. An **OrderDetail** can be queried about its ***Item***, but not the other way around. The arrow also lets you know who "owns" the association's implementation; in this case, **OrderDetail** has an **Item**. Associations with no navigability arrows are bi-directional.

The **multiplicity** of an association end is the number of possible instances of the class associated with a single instance of the other end. Multiplicities are single numbers or ranges of numbers. For example, there can be only one **Customer** for each **Order**, but a **Customer** can have any number of **Orders**.

2.4.2 Data Object Naming Conventions

(S) As shown by the matrix provided in Table 1, the standard and format for a particular name in the data domain depends on the type of data object. For example, the name of an entity or class follows the entity naming standard and the name of a field follows the attribute naming standard. This information needs to be considered when creating a data object name.

If the object is:	Subject Area Naming Standard	Entity Naming Standard	Attribute Naming Standard	Relationship Naming Standard
Subject Area	X			
Entity or Class		X		
View		X		

If the object is:	Subject Area Naming Standard	Entity Naming Standard	Attribute Naming Standard	Relationship Naming Standard
Database		X		
File		X		
Field			X	
Record		X		
Table		X		
Attribute			X	
Column			X	
Relationships				X

Table 1: Data object names matrix.

(G) Data object names should have the following properties:

- Be unique
- Have meaning to the end-user
- Contain the minimum number of words needed to uniquely and accurately describe the object
- Remain unchanged at the different data model levels
- Not be longer than 32 characters.

For entities and attributes, names are singular nouns while relationship names are typically a combination of a noun and a verb. The matrix shown in Table 2 provides more detail on how these naming standards apply to four of the more common types of data objects. The use of modifiers is optional and should be selected carefully. Object type names should be short and precise. This information needs to be considered when creating a name for a subject area, entity or class, attribute, or relationship (S).

Object Type Name	=	Component 1		Component 2		Component 3
Subject Area Name	=	Modifier(s)[5] (Optional)	+	Plural Noun (Mandatory)		
Entity or Class Name	=	Modifier(s) (Optional)	+	Singular Noun (Mandatory)		
Attribute Name	=	Noun or Noun Phrase	+	Modifier(s) (Optional)	+	Class Word (Mandatory) See Section 2.4.4, generic class words
Relationship Name	=	Noun or Noun Phrase	+	Verb	+	Noun or Noun Phrase

Table 2: Naming standards matrix.

Summarizing some key points from these naming standards:

- (S) A noun or noun phrase is used for all types of data objects except relationships.
- (S) The noun or noun phrase is singular except for a Subject Area name that has a plural or collective sense.

The naming conventions for entity modeling are:

- **(S) Entity name:** The entity name must be unique across all data models and must be singular to avoid vagueness about what a single entity occurrence represents. For example, if occurrences of an entity represent employees, name the entity "Employee" instead of the plural "Employees"; likewise, use "Student" instead of "Students". Use names that correspond to the business users' terminology. Where different groups of users have different terminology, select the more widely used or accepted term as the name for the entity. Include any synonyms for that term in the entity definition. Reflect the thing the entity represents, instead of any medium on which it is handled. For example, when naming an entity describing the event-oriented data associated with the receipt of a written complaint from a student, use something like "Student Complaint" rather than "Student Complaint Letter". The entity name does not reference the subject area as part of the name.

- **(S) Attribute name:** The attribute name must be represented with one or multiple nouns or a noun phrase and a class word. The attribute name should be unique within the EDD (for example, Social Security Number**).** If the attribute name consists of more than one word, the words are not concatenated through hyphens or underscores. At Federal Student Aid the attribute name is determined by the business term of the corresponding basic core component in the XML Registry & Repository. The attribute name does not reference an entity as part of the name. When transforming a logical data model in to a physical data model, the attributes are converted into columns. If the attribute name

[5] Modifiers can be adjectives, adverbs, absolute phrases, infinitive phrases, participle phrases, prepositional phrases, adjective clauses, and adverb clauses. For an example go to: http://www.chompchomp.com/terms/modifier.htm

consists of multiple words, these words need to be concatenated through underscores when becoming column names.

- **(S) Relationship name:** The relationship name must be a present tense verb surrounded by two entity names and can be expressed in either of two directions, depending on which entity is listed first. The name is composed of the following components through underscore:
    - Position1: entity 1 (for example: Person)
    - Position 2: verb (for example: is)
    - Position 3: entity 1 (for example: Student)

  For instance, the full name of a relationship could look like: Person is Student.

- **(S) ERD name:** The ERD name must be unique across all data models and must be singular. The name of the diagram is composed of the following components concatenated through underscore:
    - Position1: project folder (for example: name of the primary business unit the project falls under)
    - Position 2: project name (for example: the project itself, such as IPM)
    - Position 3: indicator for type of data model (for example: "c" for conceptual, "l" for logical, and "p" for physical data model)
    - Position 4 - version of the model (for example: v1)

  For instance, the name of an ERD could look like: BusinessUnit_IPM_p_v1.

2.4.3 Data Object Definitions

In addition to the general definition standards outlined in Section 2.3.2, the following additional standards apply specifically to all Data objects, except Subject Area and Relationship:

- (S) The definition must begin with a singular noun or noun phrase that makes sense as a complete sentence when preceded by "A <*data object name*> is …"
- (G) Examples are encouraged. Precede examples with the phrase "Examples include."
- (G) In a conceptual data model, a data object may not be fully defined.

2.4.4 Generic Class Words

Appendix D provides a table that describes in more detail an initial set of 22 quantitative or qualitative generic elements/class words: 6 class words are recommended for use, 16 class words are optional. Generally, class words are found as the last part for both XML business term and the business name for tabular attributes.

The following are the ***approved*** class words at Federal Student Aid:

- Quantitative: Amount, Date
- Qualitative: Code, Number, Name, Type

In addition, there are 9 ***quantitative*** generic elements/class words for optional use:

- Quantity
- Rate
- Year
- Time
- Timestamp
- Count
- Cycle
- Duration
- Percent

There are an additional 7 ***qualitative*** generic elements/class words for optional use:

- Identifier
- Period
- Indicator
- Text
- Description
- Status
- Image

(S) If an existing attribute of the legacy systems will be used to produce the future application data model development and it does not have one of the proposed generic elements/class words at the end of its name, then the attribute will be renamed. The generic element/class word that best corresponds to the purpose of the attribute will be selected as its new name. The recommended length of an attribute name is a maximum of 32 characters. If the name of the attribute is longer, the attribute name will be truncated to 32 characters when developing the physical data model, which will no longer support the goal of keeping consistent naming across the different levels of the data models.

2.5 Conceptual Data Modeling (CDM) Standards and Guidelines

These standards and guidelines determine the content, context and presentation of the information of a CDM. CDMs consist of high-level data entities and their relationships. The CDM describes key business information by subject area from a data perspective.

2.5.1 Conceptual data model packaging:

- (G) A CDM may have as many entities as necessary, but it should be divided into manageable size subject areas. In practical terms, this means a model usually has between 10 and 15 entities per subject area, with a preference toward the lower limit of 10, on a standard eight and one-half inch by eleven-inch page.
- (G) If a subject area is exceptionally large, it should be spread out on multiple pages to avoid overcrowding a single page.
- (S) Every subject area must have a unique title and a model version number.

2.5.2 Conceptual data model level of detail:

- (G) A CDM should start at a very high level, showing major entities and primary relationships. A major entity can have the same name as a conceptual data model subject area. A major entity does not inherit part of its primary key from any other entity.
- (S) No attributes may be entered in the conceptual data model.
- (G) Every major entity appearing on a conceptual data model should have at least one relationship to another entity.
- (S) Relationships appearing on an entity diagram must clearly display a verb phrase and the direction in which the action of the verb applies.
- (G) In building a CDM using ERDs, the non-specific relationship line is typically employed to model relationships between entities. Beyond the non-specific relationship line between entities, however, it is permissible to employ a more refined level of detail by using identifying or non-identifying relationships.
- (G) The verb phrase for reading the association relationship in the reverse direction may also be shown.
- (G) Every subtype relationship has a cardinality or multiplicity of "1" on each end, and implies the verb phrase "is a" in both directions. Therefore, neither this verb phrase, nor this cardinality or multiplicity, need to be entered into the model or appear on class diagrams. Global statements in data model introductions may reiterate this guideline.
- (G) Every parent/child relationship has a cardinality or multiplicity of "1" on the parent end, "zero to many" or "one to many" on the child end, and uses the verb phrase "is composed of" when read from parent to child. Therefore, neither this verb phrase, nor this cardinality or multiplicity, need to be entered into the model or appear on class diagrams. Global statements in data model introductions may reiterate this guideline.
- (S) Association relationship verbs or verb phrases must be strong, specific, active-tense in one direction and passive in the other instead of general, one-word verbs of being such as "is," "has," or "contains." It is permissible to begin verb phrases with the verbs in these examples. It is also permissible to use one-word verbs if they say something specific such as "files," "owns," or "resolves." For example the relationship between "OrganizationType" and "OrganizationTypeRole" (associative entity) can be described as:

a) " One (1) OrganizationType is allowed to act in zero (0) or many (m) OrganizationTypeRole" , and the reverse relationship reads
b) " One (1) OrganizationTypeRole is classified by one (1) OrganizationType"

2.5.3 CDM Review Template

(S) Appendix G provides a template, consisting of 9 items to be checked, to be used to review a conceptual data model for compliance with the standards and guidelines in Section2.5.

2.6 Logical Data Modeling (LDM) Standards and Guidelines

All of the above standards and guidelines that apply to a conceptual data model also apply to a LDM. In addition, the logical model has five more major data modeling standards immediately below for further refining the conceptual data model.

- (S) Compared to a CDM, a corresponding logical data model also has associative classes to resolve many-to-many relationships, is fully attributed, and is normalized to Third Normal Form (3NF). This normalization includes displaying all foreign key migration.
- (S) If a CDM was used as a foundation for adding details to develop a logical data model, then the non-specific relationship line between entities will be replaced with identifying or non-identifying relationships.
- (G) An associative entity inherits its primary key from two other entities having many-to-many relationships. For example, a conceptual data model shows only Person and Organization with an association relationship between them having cardinality or multiplicity of one or more. A corresponding logical data model also has the associative entity Person Organization, concatenating the name of both contributing entities, with a cardinality or multiplicity of exactly one.
- (S) A LDM also shows all native (that is, non-foreign key) primary key attributes and non-key attributes in the attribute area. Non-key attributes, without any stereotype identification, will be listed immediately below the primary key attributes that have a primary key stereotype.
- (S) A fully attributed logical data model will be in Third Normal Form (3NF). This means that each entity instance has exactly one unique record. All non-key attributes fully depend on primary key attributes, and no non-key attributes depend on any other non-key attributes.
- (G) Code attributes may be shown on logical data models if the code symbols and their corresponding code meanings are embedded in the user lexicon. In situations where the code values are not common knowledge, logical data models should instead use additional type entities and name attributes representing each code value.
- (S) Depending on the particular logical data modeling methodology and tool used, there are a number of acceptable ways to indicate cardinality or multiplicity on the ends of relationships between two equally important entities. This cardinality or multiplicity notation must convey one of the following meanings:
  - Exactly one
  - Zero or one

- Zero or more
- One or more

2.6.1 LDM Review Template

(S) The template in Appendix H, consisting of 27 items to be checked, will be used to review a LDM for compliance with the standards and guidelines in Section 2.6.

2.7 Physical Data Model (PDM) Standards and Guidelines

The objective for creating these standards and guidelines is to help emerging and current applications develop and maintain their physical data models in a consistent manner.

Physical database design is the process of converting the detailed logical data design into a design that can be supported by the selected database management system (DBMS). It is assumed that all target databases will be implemented in a relational database management system rather than in hierarchical, networked, object database, or any kind of sequential file designs. Examples of target DBMS tools are DB2 and Oracle.

- (S) The emerging or current application physical data model will be submitted to the EDM Team in Word format for review and register in the repository
- (S) Use of Large Object (LOB) data type columns on the PDM, such as Character large Object (CLOB) and Binary Large Object (BLOB), does not eliminate the requirement to submit a complete physical data model. This standard means that all fields composing a document stored in a single LOB data type column must map to the columns in the tables where they originate on the PDM. This mapping may be to general columns such as Field Value Text and Field Definition Form Field Name.
- (S) All LOB data type columns on a physical data model will be uniquely named and defined to specifically reflect their data storage objective. All information stored in this LOB data type column is read-only and may not be changed.
- (G) If the intent instead is to store the same information in multiple states, a more general column name and definition would be used. This requires a supporting type code column defining the particular state of the stored information.
- (S) Pure character strings, without individual pieces of data identified by name tags, will continue to use the class word Text instead of XML.
- (G) Designate a unique primary key column for every table. Do not choose an existing identifying column from a table as the primary key if the value of the column ever changes or if the properties of the column must be modified. Compound keys are even more likely to be a problem. Many database administrators include a unique numeric identifier that is generated by the database itself or by a trigger. This number should have no business meaning whatsoever. It should never change, so it is a perfect candidate for primary keys and is as simple to use as a foreign key in associated tables.
- (G) Every table should include a Timestamp column in order to manage concurrent access by multiple users if optimistic locking is enabled. Optimistic locking permits more than one user to access a table row simultaneously. The alternative is to use pessimistic locking, which prevents access to any row that is currently held by

another user. Pessimistic locking is not normally recommended for online processing, but it may be necessary for some batch processes.

- (G) Table and column names must be sized to fit the requirements of the target DBMS tool. Different database products permit different name lengths.
- (G) Table and column names should be singular.
- (G) Each column name should contain all of the elements of the logical attribute from which it was derived, but should be abbreviated to fit within the maximum length.
- (G) Most DBMS tools do not permit blanks in table or column names. Two common methods to deal with this are to insert underscore characters to replace blanks or to remove the blanks completely while capitalizing the first character of each term. Not all databases honor case sensitivity in table names, so it is preferable to separate terms with underscore characters.
- (G) Do not use hyphens in table or column names because some programming languages interpret hyphens as subtraction operators.
- (G) Implement table and column names in a way that is supported by all target DBMS tools (such as DB2, Oracle, and SQL Server).
- (S) The physical model will assign lengths and data types to all columns. Data types should be specific to the target DBMS tool. When modeling a data type not supported by the modeling tool, insert a comment to explain your intention.
- (S) The physical data model will, at a minimum, provide examples of possible values for identifier, indicator, and code columns.
- (G) A certain amount of denormalization is usually necessary when implementing the physical data model. Denormalize only if you can demonstrate a performance gain. Losses in maintaining data integrity must be justified by the performance gain. Some examples follow:
    - Replication of some attributes to reduce the number of multi-table joins needed for commonly used functions. Justification should explain how the replicated instances are to be synchronized.
    - Storage of computed values to avoid joins for aggregations. How and when will computation occur?
    - Allow a limited usage of repeating fields. For example, Phone1, Phone2, and Phone3 might be a reasonable denormalization, rather than creating a separate Phones table.
- (G) Document the range of values (domain) for each column. Some of these may be enforced by means of constraints or relationships to other tables.
- (G) Estimate the expected storage requirements for each table based on the size of each row, expected growth, number of rows, and archiving requirements.
- (G) Define alternate keys that will enhance performance by supporting common search paths.
- (G) Understand the capabilities of the specific database product. Performance improvements may be realized by taking advantage of features such as clustered indices, caching, and index optimization.

- (G) Define security requirements for every attribute and plan for implementation of security policies.
- (G) Although this document does not address the actual mechanics of converting a logical data model to a physical data model, a detailed explanation of the migration between these two data models must accompany the physical data model unless such migration is intuitively obvious to every model reviewer. This migration explanation must address the addition of new columns to existing tables, new tables that need to be part of the physical design, entities now stored in rules engines, and relationship changes.
- (G) A PDM for a current application release may display tables and columns that will not be populated until later releases if:
  - It is done for all of the columns that compose a particular table, and
  - A note identifies each unpopulated table. For this guideline, an attribute that may not be null is considered to be populated if it initially uses a default value.
- (G) A PDM may use either vertical or horizontal table design. The LOB data type appears the same way on the PDM regardless of which table design is used.

This is a choice between a larger number of smaller rows (vertical) or a smaller number of larger rows (horizontal). The vertical table design supports tax form changes from year to year. The horizontal table design may require column changes when the tax form changes.

- (G) A physical data model may actually store a document in a separate table or column that is linked to the primary LOB data type attribute.
- (G) If necessary, PDM tables may display referential integrity rules.

This optional information may provide more insight into the foreign key migration automatically shown on a PDM. Referential integrity rules must be defined for constraints on a named foreign key for updates or deletions. These rules may be restricted to set to null, or cascade.

2.7.1 PDM Review Template

(S) The template in Appendix I, consisting of 16 items to be checked, will be used to review a PDM for conformance with the standards and guidelines in Section 2.7.

3.0 Data Model Registration (DMR)

3.1 DMR Policy

It shall be the policy of Federal Student Aid that each Business Owner work diligently to comply with the Federal Student Aid ECDM for all new data systems involving core data. Each Business Owner shall provide a copy of the project-related conceptual, logical and physical data models to the EDM Team for registration. This policy applies to any major versions of the data models.

The best way to encourage data sharing and re-use is to improve awareness and understanding of data within a single Federal Student Aid Common Data Architecture (CDA).

Key policy requirements include:

- Standardizing data models (including ERDs, entities, relationships and attributes) to meet the requirements for data sharing and interoperability among information systems at Federal Student Aid.
- Using applicable standards such as ISO/IEC 11179 and Federal Student Aid's Data Standardization Policies and Procedures, Draft Enterprise Data Management Data Policies, or common best practices.

Adhering to these policies will support data management through clear, consistent, unambiguous and easily accessible data Federal Student Aid-wide and minimize the cost and time required to transform, translate, or research differently named or described data that actually represents identical data requirements.

3.2 DMR Procedures

This section describes two different DMR procedures differing based on the status of the application to which they relate: (1) registration of data models for existing applications and (2) the registration of data models of new applications.

The EDM Team will provide the standards, guidelines and procedures for developing the future systems data models and the registration in ER/Studio. The EDM Team will

- Register all major versions of the conceptual, logical and physical data models in ER/Studio, at a minimum annually.
- Assist in the reverse engineering of the physical database into a LDM using ER/Studio, if no data model exists for a legacy application. If the legacy system is a COTS application, the data model of the COTS application must be provided and registered.

3.2.1 DMR Procedures for existing applications

The EDM Team will receive and collect all data models for the existing legacy business systems in production. The business system data modelers will reverse-engineer the existing legacy systems using the ER/Studio Tool and will capture the data model in the ER/Studio format with extension *.dm1.

The EDM Team will use the following project folder structure when registering the data models in the ER/Studio Repository:

- FSA_DA_LEGACY
  - Business System 1 (for example: CSB)
  - Business System 2 (for example: IPM)
  - Business System 3 (for example: NSLDS)
  - ARCHIVE

Each sub folder contains only the most recent version of the conceptual, logical and physical data models. All previous versions will be moved to the ARCHIVE sub folder within the FSA_DA_LEGACY folder.

3.2.2 DMR Procedures for future developments in support of TSV

When a request comes to the EDM Team to support a new development project within the Federal Student Aid business systems, the EDM Team will create a new project folder under FSA_DA_MODERNIZATION in the ER/Studio Repository directory with an empty diagram-file for entity modeling. It is important to bind the EDD to this diagram to ensure consistent use of the existing data elements.

The EDM Team will provide the newly created blank diagram-file, including a copy of the EDD, to the system integrator. The system developer's data modeler will design data models based on business requirements that are in compliance with the Federal Student Aid data standardization and data modeling standards and guidelines using the existing EDD.

In cases where the development project team is unable to find the appropriate domain information to map the attributes in their business system to data elements in the EDD, the development project team will send a request to add the newly-identified data element(s) to the EDM Team following the processes outlined in the Federal Student Aid Data Standardization Policies and Procedures document.

The EDM Team will review the request and process it in accordance with the Data Standardization Policies and Procedures. Upon approval and update of the XML Registry and Repository, the new data element will be added to the EDD. The requesting team and the business unit will be notified accordingly.

During the Stage Gate reviews, the EDM Team will validate the data models of the system developer checking for compliance with relevant data modeling policies, standards, and guidelines.

Once the validation is successful, the EDM Team will load the project data models to the ER/Studio Repository following version control.

The EDM Team will register all the data models in the ER/Studio Repository using the following folder structure in the ER/Studio Repository.

- FSA_DA_MODERNIZATION
  - Business System 1(for example: IPM)
  - Business System 2 (for example: NSLDS)
  - ARCHIVE

Each sub folder contains only the most recent version of the conceptual, logical and physical data models. All previous versions of the data models will be moved to the ARCHIVE sub folder within the FSA_DA_MODERNIZATION folder.

3.2.3 DMR Process Flows

The EDM Team will capture all the data models of the legacy systems and modernization projects and it will maintain them in the ER/Studio Repository. Figure 1 below describes the high level view of the data model registration process flow.

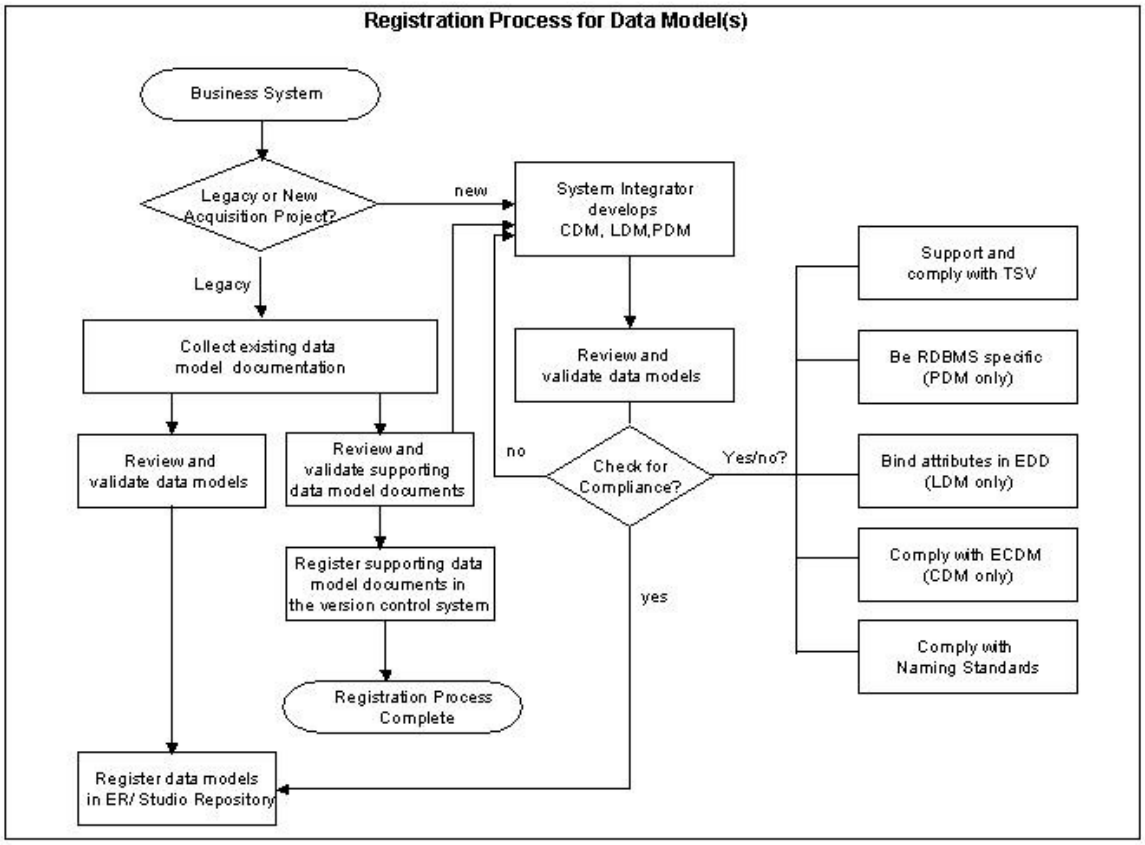


Figure 1: Registration process flow diagram.

The system developers from the business sponsor will submit their data models for registration with the EDM. The EDM Team will maintain all the data models of existing legacy systems and

on going modernization projects and will use or share this information across the enterprise as needed for the purposes of integration, migration, data exchange.

As part of the DMR Policy, the EDM Team collects data models for the following scenarios:

1. The initial registration in the repository of a business sponsor's legacy or modernization project data models.
2. Re-submission to capture major changes to data models.
3. Regular submission of the data model at least annually from the business sponsor.

The above scenarios have been documented as process flows and are enclosed in appendix O. As part of each data model submission, the system developer from the business sponsor will fill and submit the data model registration template. This template can be found in the appendix P.

Additionally, the EDM Team has developed templates for capturing entity and attribute information for a LDM, and table and column information for PDM registration. These templates are available in Appendix P. Submission of these templates is not mandatory if the system developer from the business sponsor can provide this information in a format using a data modeling tool that is compatible with ER/Studio. For PDMs, the submission of the Data Definition Language (DDL) used to generate the database is acceptable.

The EDM Team will develop and maintain the ECDM (Enterprise Conceptual Data Model). The ECDM will be available for certain business sponsors for review. For any proposed changes to this ECDM data model, the business sponsor has to follow the process as defined in appendix O and submit an ECDM change request using the template as defined in appendix P.

3.3 Roles and Responsibilities for the DMR Process

The roles and responsibilities for the DMR process described below pertain to future data model development in compliance with Federal Student Aid's data modeling standards and guidelines. The EDM Team and the system developer's data modelers have different responsibilities.

3.3.1 DMR Responsibilities of the EDM Team

The EDM Team has the following responsibilities:

1. Establish data model standards and guidelines following DoED data modeling standards.
1. Capture and register all existing legacy data models at Federal Student Aid in the ER/ Studio Repository.
2. Obtain or create existing legacy and custom application data models in the ER/ Studio format.
3. Ensure that all the future business systems data model development use the EDD for all systems development.
4. Maintain COTS product exchangeable information through creation of the data model in the ER/Studio format in the ER/Studio Repository.

5. Capture and register all data modeling supporting artifacts such as data mapping documentation and business rules.
6. Establish and maintain the Data Model Repository, which will contain the data models, following version control.
7. Review and validate the data models prior to registration (optional for COTS data models).

3.3.2 DMR Responsibilities of the System Developer

The system developer has the following responsibilities:

- Develop data models in accordance with the Federal Student Aid data modeling guidelines and procedures.
- Coordinate data model validation with the Federal Student Aid business community.
- Submit data models for stage gate review. The data model reviews are a part of the stage gate reviews. The electronic copy shall be provided to the EDM Team five business days prior to the stage gate review.

Appendix R provides standards on how to construct a well-written data definition. It also contains information about the structure and the requirements of definition including examples of good and poor data definitions.

3.4 Registration Requirements for Data-Related Artifacts

This section outlines the registration requirements for all data related artifacts such as metadata and constructs of a conceptual or logical data model as well as supporting documentation.

Federal Student Aid is collecting data models and other data-related artifacts for all system applications whether they are developed in-house or COTS packages. In any case, it is important to collect and register the meta data information for both application types.

As with all mission-critical documentation, it is important to adhere to version control. Version control allows tracking of changes to the metadata or data model over time. Any and all changes to published data models or related artifacts will be tracked and catalogued as they are made by the designated editors, utilizing commonly accepted version control techniques utilized in the software industry. Following the PESC Version Control Standards, two decimal places are utilized to allow for three categories of changes to the documentation. Using the format A.B.C, where A represents a critical change, B represents a non-critical change, and C represents micro changes such as a change to comments included in a data object. A log will be maintained within the standards file of each change, by date and author. Each data model or artifact will have a version number.

Section 3.4.4 describes the meta data registration requirements for commercial-off-the-shelf-software (COTS) products. As meta data of COTS products is often a “blackbox” for the buyer, detailed information might not be available and therefore the registration process needs to be adjusted. For instance, the data model might only be available as a pdf-file, but not in a format that can be imported to ER/Studio for further manipulation.

3.4.1 Data Model Meta Data Registration Requirements

The data model format has required metadata at the conceptual, logical, and physical data model levels. Each data model has a total of 10 mandatory or optional metadata field requirements. . These metadata fields are: Phase, Identifier, Version, Current Version, Status, Status Date, Model Creator, Steward, Subject Area(s), and Comment Text. An explanation of each of the metadata fields and information on whether the information is mandatory or not is provided in Appendix E.

3.4.2 Data Model Entity or Table Meta Data Registration Requirements

The data model entity or table format requires metadata at the conceptual, logical, and physical data model levels. As a data model entity matures and progresses through these three phases, more fields are populated in its metadata string.

Each data model entity or table has 14 mandatory or optional metadata field requirements. These meta data fields are: Name, Alias, Definition, Phase, Identifier, Version, Current Version, Status, Status Date, Model Creator, Steward, Using Model(s), Subject Area(s), and Comment Text. An explanation of each of the meta data fields and information on whether the information is mandatory or not is provided in Appendix E. A parent data model entity or table must be created before any native (that is, non-foreign key) child attribute or column can be traced to it in the Federal Student Aid EDD.

3.4.3 Attribute/ Column Meta Data Registration Requirements

The attribute or column format has required metadata at the conceptual, logical, and physical data model levels. As an attribute or column matures and progresses through these three levels, more fields are populated in its meta data string.

Each attribute or column has the same 14 mandatory or optional metadata field requirements as its parent data model entity or column plus eight additional fields unique to attributes or columns. The eight additional meta data fields are: Parent Name, Parent Identifier, Data Type, Maximum Length, Possible Values, Reference Source, Security and Privacy Requirement, and Optionality. An explanation of each of the metadata fields and information on whether the information is mandatory or not is provided in Appendix F. The composite attributes or column meta data requirements are also addressed in Appendix F. Unless otherwise noted, these meta data requirements apply to all attributes or columns regardless of whether they are quantitative or qualitative.

3.4.4 Registering COTS Meta Data Requirements

The EDM Team will maintain the COTS product exchangeable information such as “Input / Output Interfaces” meta data. If possible, this information will be captured in the data model format. COTS Metadata is often proprietary to the vendor of the system, so detailed information is not always accessible to Federal Student Aid. In some cases, meta data information is made available by the vendor in a read-only format and cannot be modified or maintained by the EDM Team or a system developer. For instance, a copy of the LDM is provided electronically in PDF format but not in a format that data modeling tools can interpret. In situations like this, the EDM Team will register the documentation in a file system outside of ER/Studio.

3.4.5 Registering all the supporting documents related to data models

The EDM Team maintains all data models in the ER/Studio Repository. ER/Studio supports only *.dm1 format data models. To register all the supporting documents (including the non-dm1 formats) related to the data models, the EDM Team creates a project folder structure in eRoom similar to that described in Section 3.1. It will use this structure to maintain all supporting documents outside of ER/Studio using version control.

Artifacts covered as part of the DMR Process:

Non COTS Application:

- All data models in the ER/Studio format should be registered in the ER/Studio repository.
- All templates of the meta data items are stored in the version controlled repository.
- All review templates are stored in a version-controlled repository (eRoom).

COTS Application:

- Data models of the COTS product are proprietary. Only part of the information (like user manuals, reference guides) is stored in the version-controlled repository (eRoom).
- All the customization of business rules for the application is captured in the version controlled repository (eRoom).
- All the information related to data exchange should be captured in the version controlled repository (eRoom).

Appendix A. Glossary

The following terms are used in this document or are pertinent to its content.

Column: A set of data values of the same type collected and stored in the rows of a table.

Database: A set of table spaces and index spaces.

Data Element: A generic term for an entity, table, attribute, or column in a conceptual, logical, and physical data model.

Enterprise Conceptual Data Model (ECDM): One of the initial components of Enterprise Data Architecture. The first enterprise level data model developed. The ECDM identifies groupings of data important to Lines of Business, Conceptual Entities, and defines their general relationships. The ECDM provides a picture of the data the enterprise needs to conduct its business. (**Reference**: *U.S. Department of Education Enterprise Data Architecture – Enterprise Data Standards and Guidelines.*)

Enterprise Logical Data Model (ELDM): A component of a maturing Enterprise Data Architecture. The second enterprise level data model developed. It is the result of merging application level data model information into the existing Enterprise Conceptual Data Model (ECDM). The ELDM extends the ECDM level of detail. (**Reference**: *U.S. Department of Education Enterprise Data Architecture – Enterprise Data Standards and Guidelines*)

eXtensible Markup Language (XML): A meta-markup language for describing data elements that is extensible because it does not have a fixed set of tags and elements.

Schema (XML): A definition, written in eXtensible Markup Language (XML) syntax, of constraints for the content type and data type of XML tags.

Schema (Data): Any diagram or textual description of a structure for representing data. (**Reference:** *FSA-EDM*)

Tag (XML): The markup portion of an Extensible Markup Language (XML) element surrounding the character data. The name of the tag reflects the content inside the XML element.

Appendix B. Abbreviations / Acronyms

The following abbreviations and acronyms are used herein or are pertinent to content included herein:

Abbreviation / Acronym	Applicable Term
CDM	Conceptual Data Model
DBMS	Data Base Management System
DMR	Data Model Registration
ECDM	Enterprise Conceptual Data Model
ED	Department of Education
EDD	Enterprise Data Dictionary
EDM	Enterprise Data Management
ELDM	Enterprise Logical Data Model
ERD	Entity Relationship Diagram
FEA	Federal Enterprise Architecture
FEAF	Federal Enterprise Architecture Framework
FIPS	Federal Information Processing Standards
IEC	International Electro-technical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
LDM	Logical Data Model
PDM	Physical Data Model
PESC	Postsecondary Electronic Standards Council
SCM	Software Configuration Management
XML	eXtensible Markup Language
XML R & R	XML Registry and Repository

Appendix C. References

- ED Enterprise Data Standards and Guidelines – March 23, 2005
- ED Enterprise Data Management – Data Model Registration Process – January 2006
- Entity Relationship Modeling Techniques:

  Concepts – http://sysdev.ucdavis.edu/WEBADM/document/td_entityrel-concepts.htm

  Rules - http://sysdev.ucdavis.edu/WEBADM/document/td_entityrel-rules.htm
- ER/Studio Repository Documentation – version 7.0.1
- Introduction to Data Modeling (URL: http://www.utexas.edu/its/windows/database/datamodeling/dm/erintro.html)
- Federal Student Aid – 15.025 Draft Enterprise Data Management Data Policies, June 1, 2007
- Federal Student Aid - Data Standardization Policies and Procedures, Version 1.0 – May 2007
- Federal Student Aid – Enterprise Data Dictionary Standards, Version 1.0 – April 2007

Appendix D. Generic Elements/ Class Words

List of 6 approved class words for the Federal Student Aid to be used for each attribute or column name.

Generic Element/Class Word Name	Definition	Abbreviation	Typical Feeder Terms Represented	Metadata Field Parameters
Amount	A monetary value.	AMT	Average, Balance, Cost, Price	Defined "The (optional modifiers) amount of…" Integers, Fixed Point with decimal place count quantity
Date	A specific period of time described as any combination of month, day, and year.	DT	Calendar, Ordinal	Defined "The (optional modifiers) date of/when/on which…"
Code	A combination of one or more numbers, letters, or special characters substituted for a specific meaning.	CD	Category, Status, Type	Defined "The code that represents/ Denotes…"
Number	An identifier that may have nonnumeric characters; an ID number. (Do not use this to represent a count or quantity.)	NUM		Defined "The (optional modifiers) number that identifies…" The preferred class word is Identifier with Number as an optional class word modifier.
Name	A designation of an object or data model class expressed in a word or phrase.	NM	Name, Title	Defined "The (optional modifiers) name of…"
Type	A particular kind, class, group, or category	TYP		Defined "The (optional modifiers) type of…" The preferred class word is Code with Type as an optional class word modifier.

List of 17 optional proposed class words for the Federal Student Aid to be used for each attribute or column name.

Generic Element/Class Word Name	Definition	Abbreviation	Typical Feeder Terms Represented	Field or Metadata Parameters
Quantity	A non-monetary numeric value that does not have to be a whole number.	QTY	Count, Index, Mean, Median, Mode	Defined "The (optional modifiers) quantity of…" Integers, optional Floating Point or Fixed Point with decimal place count quantity
Rate	A quantitative expression that represents the numeric relationship between two measurable units.	RT	Factor, Percent	Defined "The (optional modifiers) rate of…" Integers, Floating Point or Fixed Point with decimal place count quantity
Year	A numeric value for a calendar or fiscal year.	YR	Calendar Year, Tax Year, Fiscal Year	Defined "The (optional modifiers) year of/when/in which…" Integers
Time	A notation of a specified chronological point within a period of time.	TM	Time	Defined "The (optional modifiers) time of/when…" Integers
Timestamp	A specific point of time, described as a sequential seven-part value, presented in the order of most significant component to least significant component.	TS	Timestamp	Defined "The (optional modifiers) timestamp of/when…" Format is YYYYMMDDHHMM SSNNN where: YYYY is the year; MM is the month; DD is the day; HH is the hour; MM is the minute; SS is the second; NNN is the microsecond
Count	A non-monetary numeric value expressed in integers with optional floating point or fixed point with decimal place quantity.	CNT		Defined "The (optional modifiers) count of/that…" The preferred class word is Quantity with Count as an optional modifier.
Cycle	An alternative expression of Date	CYC		Defined "The (optional modifiers)

Generic Element/Class Word Name	Definition	Abbreviation	Typical Feeder Terms Represented	Field or Metadata Parameters
	specifically used to identify file processing updates and subsequent data warehouse extractions.			cycle of/when…" Format is YYYYNN where YYYY is the calendar year and NN is the weekly cycle number
Duration	A continuous length of time expressed in a unit of measure such as milliseconds	DUR		Defined "The (optional modifiers) duration of/when/between…" The preferred class word is Quantity with Duration as an optional modifier and a specified unit of measure.
Percent	The ratio of part of whole expressed in hundredth	PCT		Defined "The (optional modifiers) Percent of…" The preferred class word is Rate with Percent as an optional class word modifier.
Identifier	A combination of one or more numbers, letters, or special characters that designates a specific object or data model class but has no readily definable meaning.	ID	Designator, Key, Number	Defined "The (optional modifiers) identifier that represents…"
Period	A range of dates and/or time that has a starting point and an ending point.	PRD		Defined "The (optional modifiers) period of/when…"
Indicator	An attribute having two possible values that are contrary.	IND		Defined "The (optional modifiers) indicator for/of…"
Text	An unformatted character string generally in the form of words.	TXT	Comment, Description	Defined "The (optional modifiers) text of…"
Description	An unformatted character string, generally in the form of words.	DSC		Defined "The (optional modifiers) description of…" The preferred class word is Text with Description as an optional class

Generic Element/Class Word Name	Definition	Abbreviation	Typical Feeder Terms Represented	Field or Metadata Parameters
				word modifier.
Status	The condition of an object or data model class at one stage in a related series of events	STAT		Defined "The (optional modifiers) status of…" The preferred class word is Code with Status as an optional class word modifier.
Image	The use of a Binary Large Object (BLOB) to store binary data such as pictures	IMG	An example of an Image would be the electronic signature on a tax return.	Defined "The picture of…" Recommend use of modifiers with the class word Image to specifically describe what the image is capturing (e.g., Electronic Signature Image).

As a general rule, fewer proposed generic elements/class words are preferable because they provide a tighter enterprise data structure and minimize potential confusion. For example, the Department of Defense (DoD) uses only four proposed qualitative generic elements/class words: Identifier, Code, Name, and Text. Department of Education deprecated the Number, Type, Count, Duration, Percent, Description, Status class words.

However the usage of the class words is varies organization to organization.

Appendix E. Entity or Table Metadata Requirements

The following abbreviations are used when describing the entity or table metadata requirements.

Requirement Indicator	Data Model Level
M = Mandatory	C = Conceptual
R = Recommended	L = Logical
O = Optional	P = Physical

The following abbreviations are used in pairs in the third, fourth, and fifth columns from the left to depict the mandatory, recommended, or optional nature of a metadata field for a data model entity or table at the conceptual, logical, or physical data model level.

Metadata Field Item	Purpose	C	L	P	Metadata Field Parameters
Name	The standardized entity name on the conceptual or logical data model, or the abbreviated table name on the physical data model.	M	M	M	
Alias	If such a name exists, the vernacular version of any standardized or abbreviated name.	O	R	M	
Definition	The standardized definition of an entity or table.	M	M	M	
Phase	The conceptual, logical, or physical state of an entity or table.	M	M	M	“Conceptual,” “Logical,” or “Physical”
Identifier	A unique, system-generated record number assigned to the entity or table.	M	M	M	A five-digit number beginning with leading zeroes and ending with a zero (e.g., 00010 for the first entity or table).
Version	A system-generated configuration management identifier to maintain the integrity of a particular metadata set, or field string, for an entity or table.	M	M	M	Maximum of two sequential integers beginning with “1”.
Current Version	A Boolean “Yes” or “No” flag indicating if this version is the recognized as the latest version of the entity or table. This flag allows only Current Version data model information to be displayed in filtering, sorting, or keyword searching.	M	M	M	“Y” or “N”

Metadata Field Item	Purpose	C	L	P	Metadata Field Parameters
Status	A stage of a particular version of a data model entity or table within the data standardization approval cycle.	M	M	M	One of five possible stages: Developmental, Candidate, Approved, Archived, Rejected. See further Status explanation immediately below this table.
Status Date	The date that the current Status became effective.	M	M	M	
Model Creator	The name of the person who originally created the data model using the entity or table.	M	M	M	
Steward	The project currently assigned data model entity or table development responsibility to	O	R	M	
Using Model(s)	The names of all approved data models that use this version of the entity or table.	M	M	M	
Subject Area(s)	The names of the subject areas within the using models displaying the entity or table.	O	O	O	
Comment Text	Any free form text that may add value to the existing metadata field string.	O	O	R	Examples of possible entries include previous entity or table approval history, mapping information, or further definition discussion to shorten the actual definition entry.

There are five potential status entries. These are *Developmental*, *Candidate*, *Approved*, *Archived*, and *Rejected*. Typically, a data element version successively moves through the first four statuses during its data life cycle.

A ***Developmental*** status data element is the initial stage of data element development. It may be used to satisfy a new data requirement but with the understanding that the data element is not yet mature.

An ***Approved*** status data element is the authoritative data source for enterprise-wide use. It must be used in its present format if it will fully satisfy a new data requirement. If it will not satisfy such a requirement, a modified version of the data element must be submitted for approval.

A ***Candidate*** status data element is the second most authoritative data source for enterprise-wide use. It can be used in its present format with a high degree of confidence. In comparison, a ***Developmental*** status data element may be used, but with the understanding that it is not yet mature.

An ***Archived*** status data element should be considered purely as a historical reference. It satisfied an earlier data requirement but should not be recreated at this time.

Finally, a ***Rejected*** status data element should not be considered at all because it never adequately satisfied a data requirement.

Appendix F. Attribute or Column Metadata Requirements

The following abbreviations are used in pairs in the third, fourth, and fifth columns from the left to depict the mandatory, recommended, or optional nature of a metadata field for an attribute or column at the conceptual, logical, or physical data model level. For example, the abbreviation combination "O" in the "C" column = Optional at the conceptual data model level. The abbreviation table is in Appendix E.

Metadata Field Item	Purpose	C	L	P	Metadata Field Parameters
Name	The standardized attribute name on the conceptual or logical data model, or the abbreviated column name on the physical data model.	M	M	M	
Alias	If such a name exists, the vernacular version of any standardized or abbreviated name.	O	R	M	
Definition	The standardized definition of an entity or table.	M	M	M	Begins with prescribed wording depending on the particular generic element word at the end of the attribute or column name.
Phase	The conceptual, logical, or physical state of an attribute or column	M	M	M	"Conceptual," "Logical," or "Physical"
Identifier	A unique, system-generated record number assigned to the attribute or column	M	M	M	For a child attribute or column, populate this column with a two-digit extension of the parent entity or table identifier (e.g., 00010.01). This extension associates the parent and the child.
Version	A system-generated configuration management identifier to maintain the integrity of a particular metadata set, or field string, for an attribute or column.	M	M	M	Maximum of two sequential integers beginning with "1".
Current Version	A Boolean "Yes" or "No" flag indicating if this version is the recognized as the latest version of the attribute or column. This flag allows only Current Version data model information to be displayed in filtering, sorting, or keyword searching.	M	M	M	"Y" or "N"
Status	A stage of a particular version of a data model attribute or column within the data standardization approval cycle.	M	M	M	One of five possible stages: Developmental, Candidate, Approved, Archived, Rejected. See further Status explanation immediately below this table.
Status Date	The date that the current Status became effective.	M	M	M	

Metadata Field Item	Purpose	C	L	P	Metadata Field Parameters
Model Creator	Person who created the data model using the attribute or column.	M	M	M	
Steward	The project currently assigned data model attribute or column development responsibility.	O	R	M	
Using Model(s)	The names of all approved data models that use this version of the attribute or column.	M	M	M	
Subject Area(s)	The names of the subject areas within the using models displaying the attribute or column.	O	O	O	
Comment Text	Any free form text that may add value to the existing metadata field string.	O	O	R	Examples of possible entries include previous attribute or column approval history, mapping information, or further definition discussion to shorten the actual definition entry.
Parent Name	The name of the entity or table to which this attribute or column belongs.	M	M	M	
Parent Identifier	The identifier assigned to the entity or table to which this attribute or column belongs.	M	M	M	
Data Type	The name of the way that the attribute or column domain values would be stored in a database; examples include character string, integer, fixed point, and floating point.	O	R	M	The specific nomenclature depends on the target database management system (DBMS).
Maximum Length	Unless already included in the Data Type, the maximum permitted field length for the attribute or column.	O	R	M	Integers
Possible Values	Paired code symbols and corresponding values or indicator values.	O	R	M	If the list of Possible Values is lengthy, provide a few examples and refer to a reference table containing all Possible Values.
Reference Source	The official regulation or policy that specifically requires the attribute or column and contains its Possible Values.	O	R	M	
Security and Privacy Requirement	Any special security or privacy requirement affect-ing this attribute or column.	O	R	M	
Optionality	The null or not null status of the attribute or column.	O	R	M	“Mandatory” or “Optional”.

Appendix G. Conceptual Data Model Review Template

Number	Standard	Status	Review Methodology	Model Discrepancies
1	Is the conceptual data model presented in a Word report format in addition to the format produced by the specific data modeling notation and tool format selected by the data modeler?	Mandatory	Visual and electronic check of the conceptual data model; negative answer is basis for returning model to project.	
2	Does the entire conceptual data model, and any subset subject area views within it, comply with the recommended size limit of 10 to 15 entities per 8.5" x 11" page?	Optional	Count of the number of entities per page	
3	Does the entire conceptual data model, and any subset subject area views within it, have a discrete title as well as a model version number?	Mandatory	Visual and electronic check of the conceptual data model	
4	Does the conceptual data model have at least one native attribute for each major entity?	Optional	Visual and electronic check of the conceptual data model	
5	Does the conceptual data model show at least one relationship to another entity from each major entity?	Optional	Visual and electronic check of the conceptual data model	
6	Are the conceptual data model relationships named in the form of a verb phrase?	Mandatory	Visual and electronic check of the conceptual data model	
7	Are the relationship name verbs or verb phrases strong, specific, active-tense in one direction, instead of general, one-word verbs of being?	Mandatory	Check of the conceptual data model to identify inappropriate, one-word verbs such as "is," "has," or "contains."	
8	If the conceptual data model relationship displays a reciprocal verb phrase, is this second verb phrase clearly understandable?	Optional	Check conceptual data model multiplicity notation.	
9	Do the conceptual data model relationships note cardinality or multiplicity on both ends of the relationship line?	Optional	Visual and electronic check of the conceptual data model	

Appendix H. Logical Data Model Review Template

Number	Standard	Status	Review Methodology	Model Discrepancies
1	Is the logical data model presented in a Word report format in addition to the format produced by the specific data modeling notation and tool format selected by the data modeler?	Mandatory	Visual and electronic check of the logical data model; negative answer is basis for returning model to project.	
2	Does the entire logical data model, and any subset subject area views within it, comply with the recommended size limit of 10 to 15 entities per 8.5" x 11" page?	Optional	Count of the number of entities per page	
3	Does the entire logical data model, and any subset subject area views within it, have a discrete title as well as a model version number?	Mandatory	Visual and electronic check of the logical data model	
4	Does the logical data model, or any subset subject area views, display all entities including independent entities, dependent or attributive entities, associative entities, and subtypes?	Mandatory	Check consists of answering more detailed questions. Does the logical data model display all applicable entities in the current ECDM? Has a sufficient number of new entities been added to attain Third Normal Form (3NF)?	
5	Does the logical data model provide appropriate rationale for the omission of any in-scope major relationships or any major entities in the current ECDM?	Mandatory	Check of the logical data model and any supporting documentation accompanying it, such as a cover letter; negative answer is basis for returning model to project.	

Number	Standard	Status	Review Methodology	Model Discrepancies
6	Does the logical data model provide appropriate rationale for changing any approved modeling constructs in the current ECDM?	Mandatory	Check of the logical data model and any supporting documentation; negative answer is basis for returning model to project.	
7	Is the logical data model fully attributed, showing all native primary keys and non-keys?	Mandatory	Check consists of verifying that the model sufficiently attributed to be in Third Normal Form (3NF).	
8	Does the logical data model show all primary keys in context entities?	Mandatory	Check logical data model against the current ECDM.	
9	Do the logical data model attribute names end with proposed class words?	Mandatory	Check attribute class words against the list of proposed class words.	
10	Does the logical data model define all entities and attributes?	Mandatory	Check all metadata in the logical data model describing each entity and attribute. Negative answer is basis for returning model to project.	
11	Do the context class and attribute definitions in the logical data model correctly?	Mandatory	Check these data element definitions.	
12	Do the new native entity and attribute definitions concisely tell what the data element is instead of how it is used?	Mandatory	Check length of data element definitions. Consider placing any “How” explanation in the comments field.	
13	Do the logical data model entities and attributes have entries in all mandatory metadata fields?	Mandatory	Check all metadata in the logical data model describing each entity and attribute against the standards and guidelines. Negative answer is basis for returning model to project.	

Number	Standard	Status	Review Methodology	Model Discrepancies
14	Does the logical data model provide appropriate rationale for any new entities or attributes that appear to duplicate existing data elements?	Mandatory	Check the logical data model, and any supporting documentation accompanying it, such as a cover letter.	
15	Does the logical data model include abbreviated class and attribute names following an authorized abbreviation list?	Optional	Check all abbreviations against list of authorized abbreviations in the standards and guidelines. Use of abbreviations is optional in the logical phase but, if present, they must match the authorized list.	
16	Does the logical data model identify attribute data types?	Optional	Check all metadata in the logical data model describing each attribute (e.g., character string, integer, fixed point, floating point).	
17	Does the logical data model list aliases for entities and attributes?	Optional	Check all metadata in the logical data model describing each entity and attribute.	
18	Is the logical data model independent of physical constraints imposed by a specific application or the target Database Management System (DBMS)?	Mandatory	Check all metadata in the logical data model to identify and remove any attributes that are purely physical, application unique, duplicate, or derived.	
19	Does the logical data model show all relationships?	Mandatory	Check the logical data model to determine if it shows all applicable relationships in the current ECDM.	

Number	Standard	Status	Review Methodology	Model Discrepancies
20	Are the logical data model relationships named in the form of a verb phrase?	Mandatory	Check wording for all logical data model relationships.	
21	Are the relationship name verbs or verb phrases strong, specific, active-tense in one direction instead of general, one-word verbs of being?	Mandatory	Check of the logical data model to identify inappropriate, one-word verbs such as “is,” “has,” or “contains.”	
22	If relationships display a reciprocal verb phrase, is this second verb phrase clearly understandable?	Optional	Check logical data model multiplicity notation.	
23	Do the logical data model relationships note cardinality or multiplicity on both ends of the relationship line?	Mandatory	Check logical data model multiplicity notation.	
24	Does the logical data model have associative classes to resolve many-to-many relationships?	Mandatory	Check logical data model for new associative entities that resolve many-to-many relationships into a pairs of one-to-many relationships.	
25	Does the logical data model include statistics on object or data model class and relationship occurrences?	Optional	If provided, these statistics will be accepted and reviewed for the sole purpose of gaining more insight into the logical data model. Statistics could be provided in supporting documentation, such as an attachment to a cover letter.	

Number	Standard	Status	Review Methodology	Model Discrepancies
26	Does the logical data model include preliminary referential integrity rules?	Optional	Such rules will be updated in the physical data model. If a specific rule is provided (i.e., restrict, cascade, or set to nulls), it will be accepted and reviewed for the sole purpose of gaining more insight into the logical data model. A fourth, default rule will be "not specified."	
27	Does the logical data model show all foreign key migration as part of normalization?	Mandatory	Check logical data model for attributes with foreign key notation.	

Appendix I. Physical Data Model Review Template

Number	Standard	Status	Review Methodology	Model Discrepancies
1	Is the physical data model presented in a Word report format in addition to the format produced by the specific data modeling notation and tool format selected by the data modeler?	Mandatory	Visual and electronic check of the physical data model; negative answer is basis for returning model to project.	
2	Does the entire physical data model, and any subset subject area views within it, comply with the recommended size limit of 10 to 15 tables per 8.5" x 11" page?	Optional	Count of the number of tables per page	
3	Does the entire physical data model, and any subset subject area views within it, have a discrete title as well as a model version number?	Mandatory	Visual and electronic check of the physical data model	
4	Are all primary key columns identified on the physical data model?	Mandatory	Check physical data model columns for this notation. Some primary key columns may also be foreign keys.	
5	Does the physical data model use the abbreviations found in standards and guidelines as required by DBMS space constraints for table and column names?	Mandatory	Check of the physical data model. This check may be automated.	
6	If the migration from the logical data model to the physical data model is not intuitively obvious, is there also a detailed explanation of this migration accompanying the physical data model?	Mandatory	Check of the physical data model, the logical data model, and any supporting documentation; negative answer is basis for returning model to project.	
7	If provided, does this migration explanation address the addition of new columns to existing tables?	Mandatory	Comparison of the physical data model, the logical data model, and the migration explanation.	
8	If provided, does this migration explanation address new tables that need to be part of the physical design?	Mandatory	Comparison of the physical data model, the logical data model, and the migration explanation.	

Number	Standard	Status	Review Methodology	Model Discrepancies
9	If provided, does this migration explanation address entities that are now stored in rules engines instead of on the physical data model?	Mandatory	Comparison of the physical data model, the logical data model, and the migration explanation.	
10	If provided, does this migration explanation address relationship changes?	Mandatory	Comparison of the physical data model, the logical data model, and the migration explanation.	
11	Does the physical data model identify all tables and columns that will not be populated until later application releases?	Optional	Check of the physical data model.	
12	Are all Large Object (LOB) data type columns on the physical data model uniquely named and defined to specifically reflect their data storage objective?	Mandatory	Check of all Large Object (LOB) data type column names and definitions.	
13	Do the physical data model tables display needed referential integrity rules for foreign key update or deletion?	Optional	Check physical model for referential integrity information. The physical data modeler determines the need for these rules.	
14	Does the physical data model identify column data types?	Mandatory	Check all metadata in the physical data model describing each column (e.g., character string, integer, fixed point, floating point).	
15	Does the physical data model identify maximum lengths for column values?	Mandatory	Check all metadata in the physical data model describing each column.	
16	Does the physical data model identify possible values for identifier, indicator, and code columns?	Mandatory	Check all metadata in the physical data model describing each identifier, indicator, and code column.	

Appendix J. Information Engineering (IE) Notation

Federal Student Aid has adopted IE[6] Notation for data modeling. Information Engineering (IE) is considered to be semantically stronger than IDEF1X, supporting more abstract approaches to data modeling. IE is more flexible in supporting subtypes, permitting roll up (generalization) and roll down (specialization) transformations. It is a streamlined refinement of the ER-modeling theme discarding the arbitrary notion of the complex "relationship" modeling them as associated entities. Thus every relationship in IE is binary, involving two entities (or possibly only one if reflexive). IE also simplified the graphic notation in the diagram style. In addition, primary keys are less prominent in the IE notation; therefore, its practitioners are less likely to think about logical key choice prematurely. The hallmark of IE notation is its "crow's feet" relationships.

IE (Crow's Feet) - Crow's Feet is a variation of the traditional James Martin notation. It uses dashed lines for non-identifying relationships and solid lines for identifying relationships. Identifying relationships are signified by rounded corners in the child table. For non specific relationships, the two entities have many-to-many relationships with an association relationship between them and a cardinality or multiplicity of one or more.

[6] Entity Relationship Modeling Technique: http://sysdev.ucdavis.edu/WEBADM/document/td_entityrel-rules.htm and http://sysdev.ucdavis.edu/WEBADM/document/td_entityrel-concepts.htm

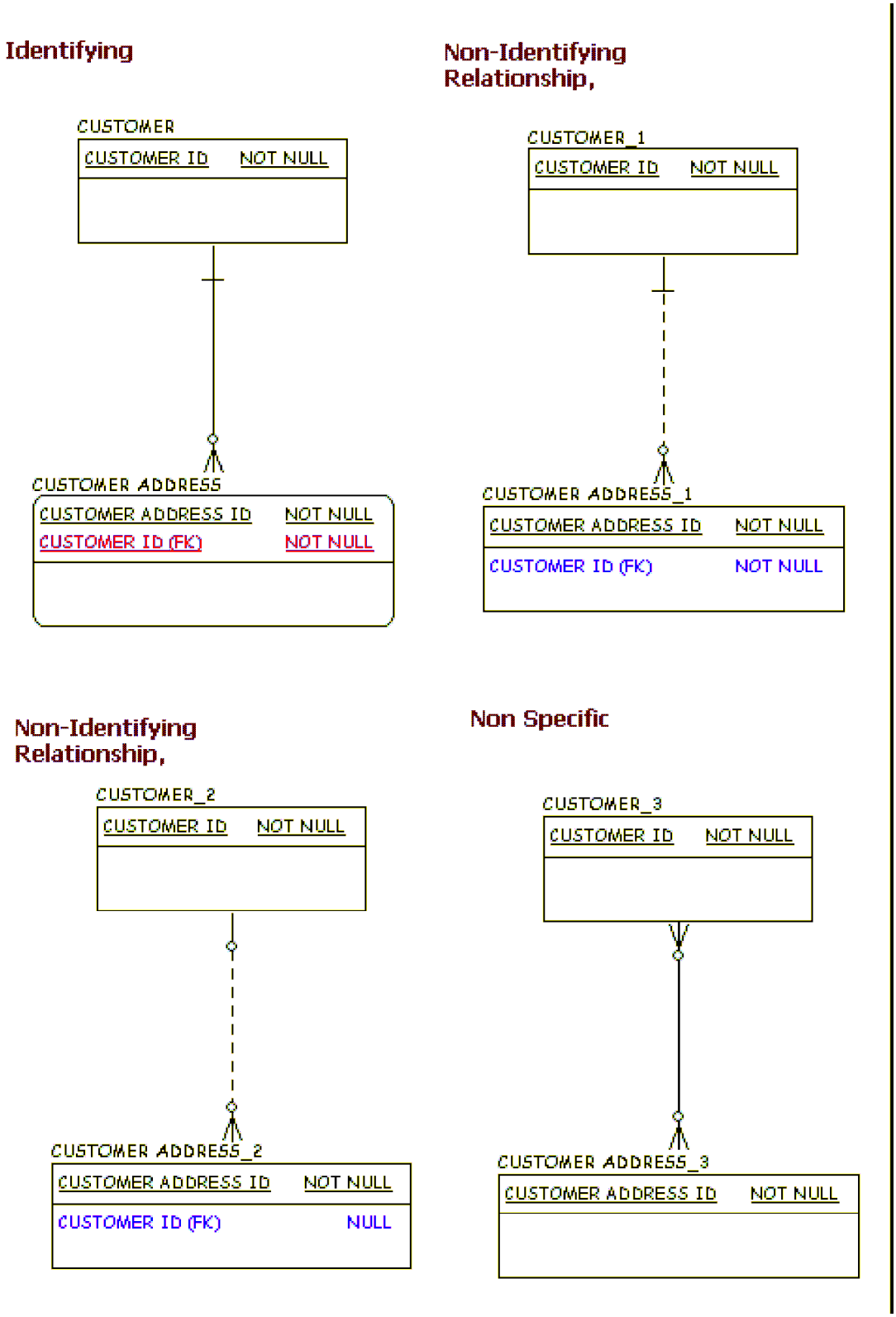


Figure J-1: IE notation.

Appendix K. ER/Studio Enterprise Data Dictionary (EDD)

ER/Studio's Data Dictionary system is a powerful aid allowing a data architect or modeler to enforce standards, promote reuse, and build a common framework across all models. A well-defined data dictionary prevents constant reinventing of the wheel and provides everything that is needed. Domains used in conjunction with reference values, defaults, and rules let the user build a common list of data elements that can be used in any logical or physical model. The attachment system allows to add additional metadata to any object in a logical or physical model. Every object editor includes an Attachment Bindings tab to add attachments.

Once the objects are defined in the Data Dictionary, they can be used in either the logical or physical models. Define domains to create "template" or reusable attributes and columns. For example a domain, ***Timestamp***, can be created for attributes or columns tracking a modify date. Now any ***LastModifyDate*** attribute or column can be bound to the ***Timestamp*** domain so that they can all be managed through that single domain. Reference values, defaults, and rules can be implemented directly on attributes and columns, or used to further define the definition of a domain. Using the previous example, a common default for ***LastModifyDate*** is the current system date. So a default can be created in the Data Dictionary called ***CurrentSystemDate*** with a different definition depending on the target DBMS. If this default is then bound to the ***Timestamp*** domain under the default tab, every attribute or column that is bound to the domain will now get a default of the current system date.

Use Data Dictionaries across many *.dm1 files. If the Repository is in use, the Enterprise Data Dictionary system can be used to implement one dictionary across disparate models. Changes to an object in an Enterprise Data Dictionary will be reflected in any model where that object is used.

Macros can also be used to import and export domain definitions and reference values from Excel if they need to be imported from other sources such as an external metadata repository.

Enterprise Data Dictionary

The Enterprise Data Dictionary (EDD) allows to share a single data dictionary among multiple diagrams in the same repository. Instead of being stored in a particular DM1 file, the EDD is stored in the repository, and "bound" to the DM1 files that use it. A change made to the EDD propagates to any DM1-object to which the dictionary is bound.

The Enterprise Dictionary Binding Dialog includes attachments and reference values. The bindings are presented in a grid format and can be exported to *.csv files for reporting purposes.

More detailed information on the EDD can be found in the “Enterprise Data Dictionary Standards, Version 1.0” document.

Appendix L. ER/Studio Repository

For better registration and maintenance, the EDM Team decided to maintain all the data models in ER/Studio Repository. This can be streamlined by creating the following folder structure before creating or adding data model diagrams.

After understanding the current requirements of the data model registration process, the EDM Team will create two folders under the projects folder in the ER/Studio Repository using the following wizard.

Go to the ER/Studio Repository, open the Repository menu, and then click Project Center (see Figure L-1).

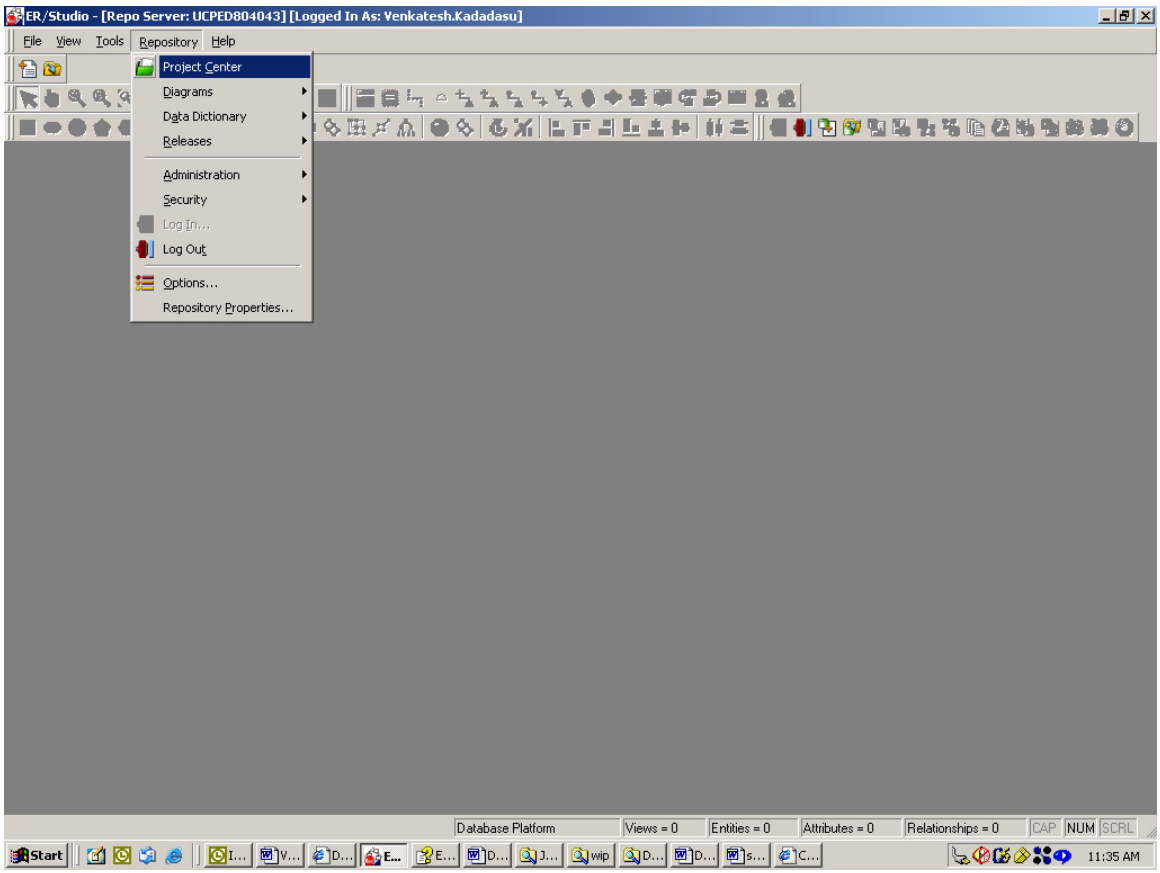


Figure L-1: ER/Studio Repository menu.

The window shown in Figure L-2 appears for creating the folder structures under the projects.

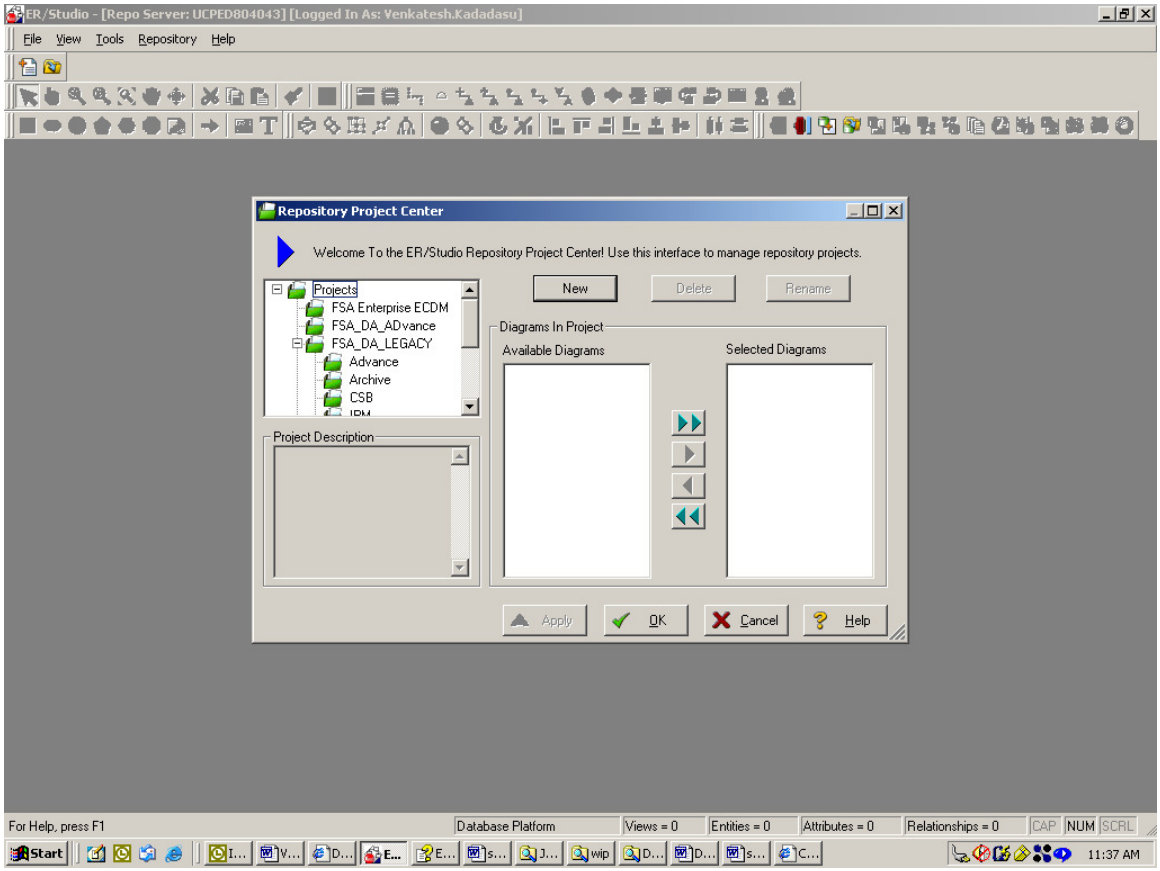


Figure L-2: Creating folder structures under the projects.

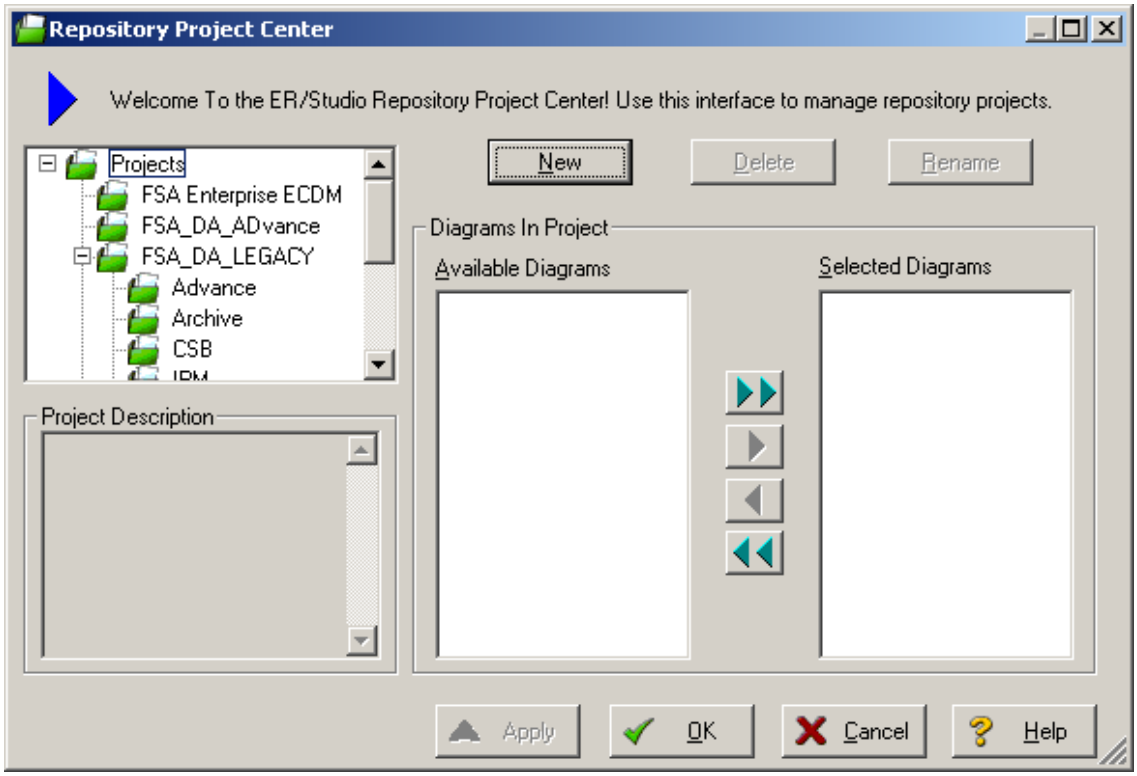


Figure L-3: Managing repository projects.

Click the New Tab to create the folder. The wizard shown in Figures L-4 and L-5 will appear to create the project names (these are similar to folder creation).

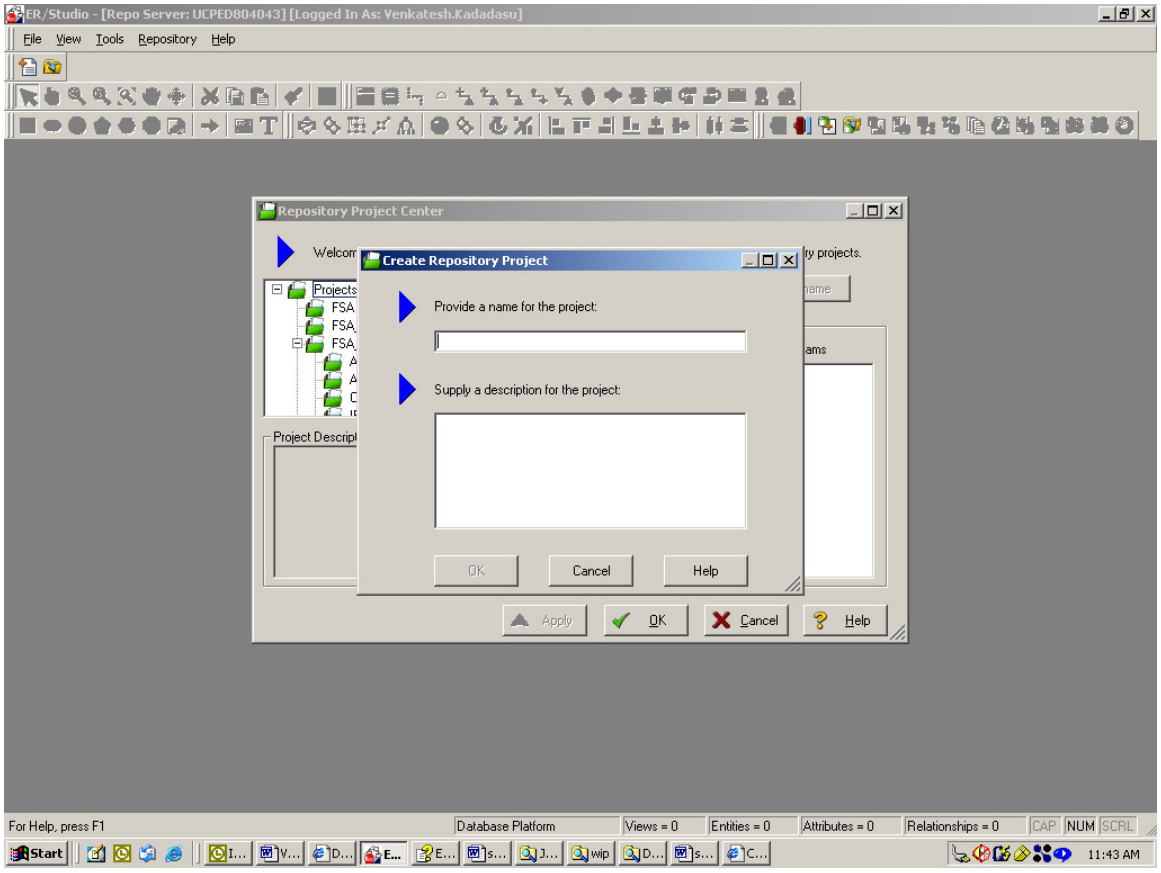


Figure L-4: The Wizard to create project names.

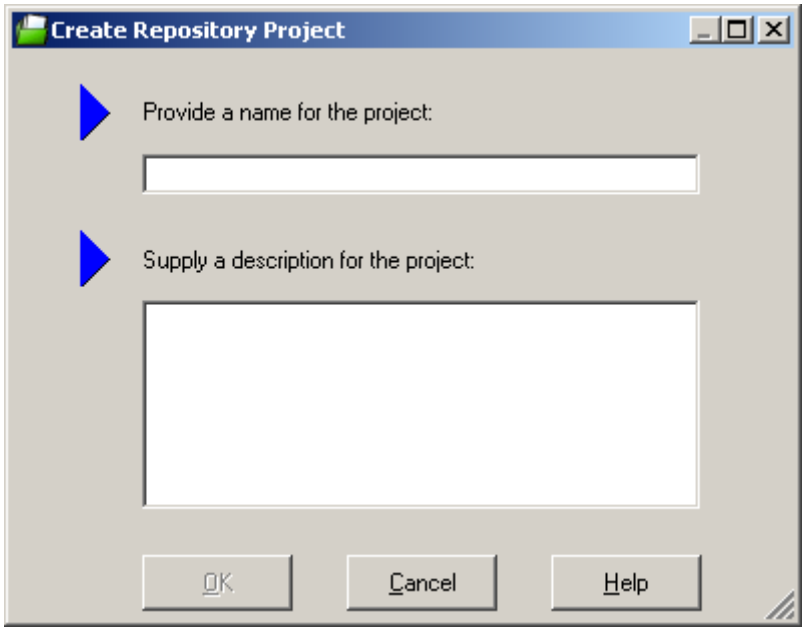


Figure L-5: Close-up of the Wizard's dialog box.

Enter the project name and brief description in the dialog box, as shown in Figures L-6 and L-7.

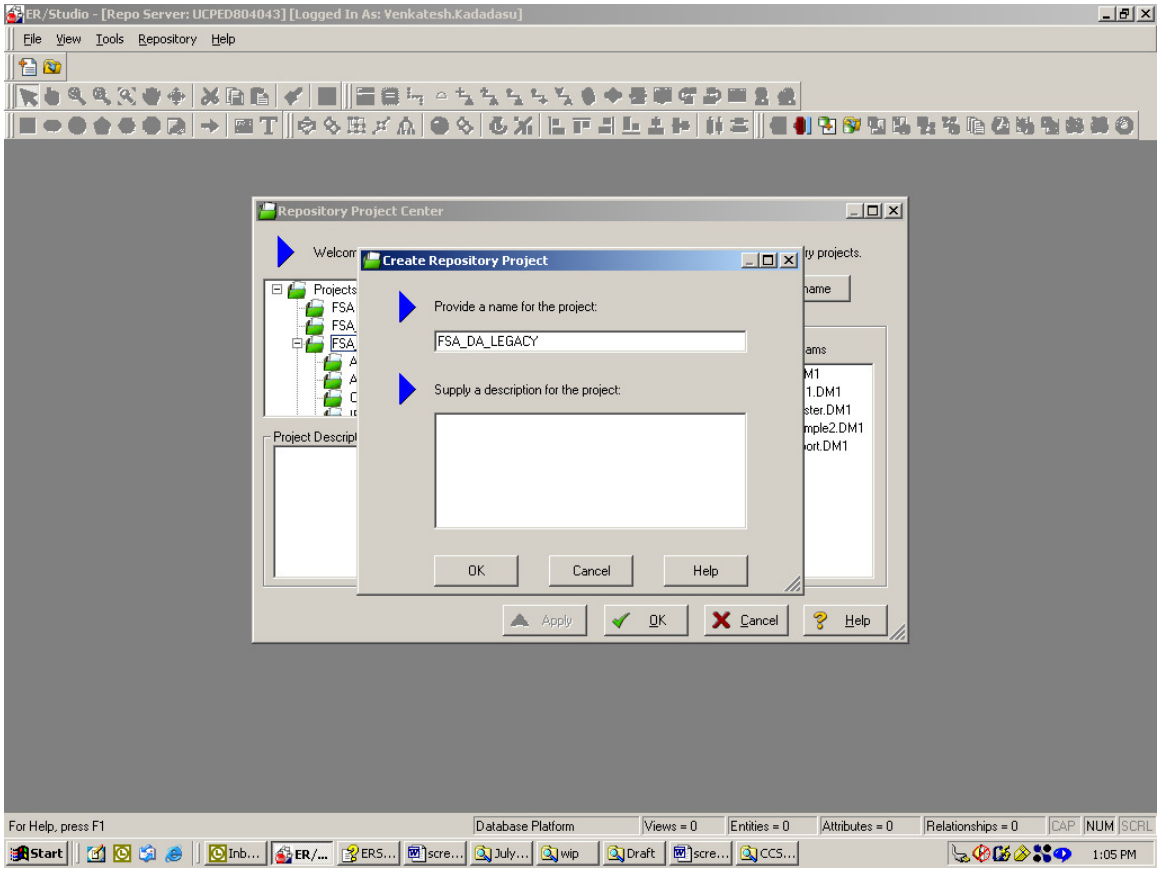


Figure L-6: Entering the project name and description.

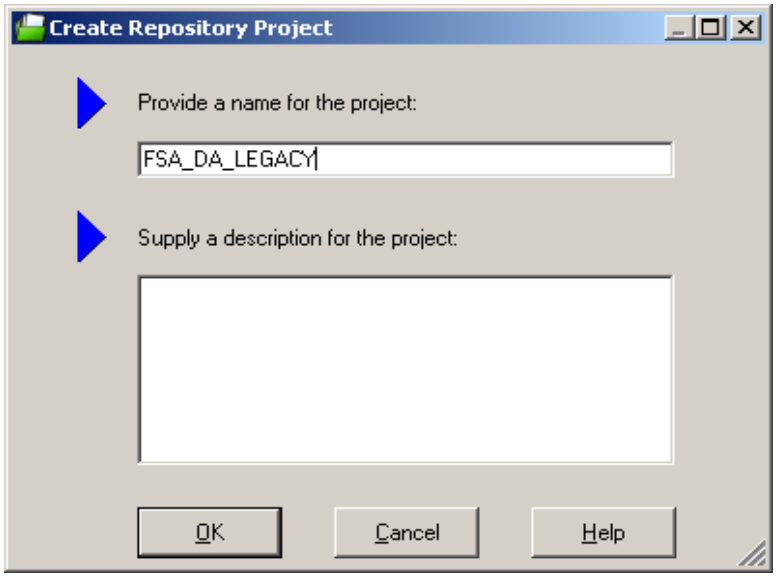


Figure L-7: Close-up of entering the project name and description.

This will create the new project called "FSA_DA_LEGACY" under the Projects in the Repository Project Center.

After entering the project name and description, click OK.

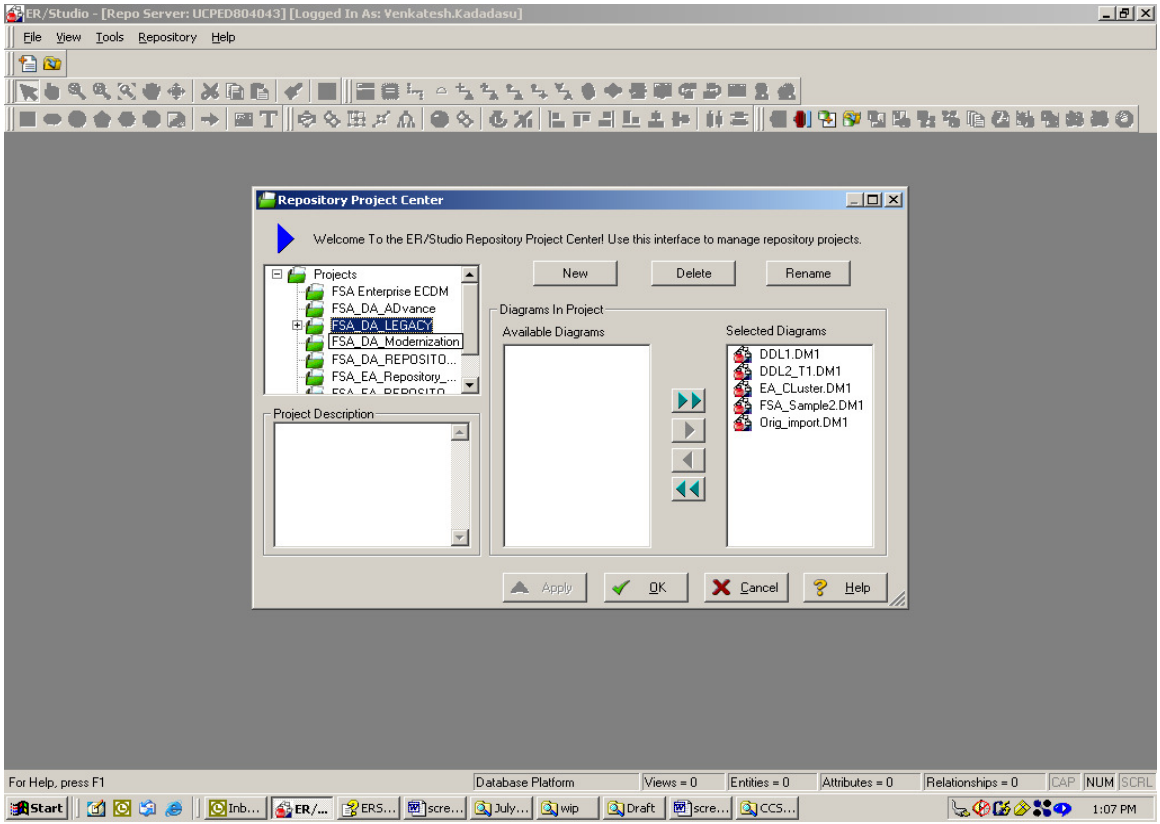


Figure L-8: The Repository Project Center.

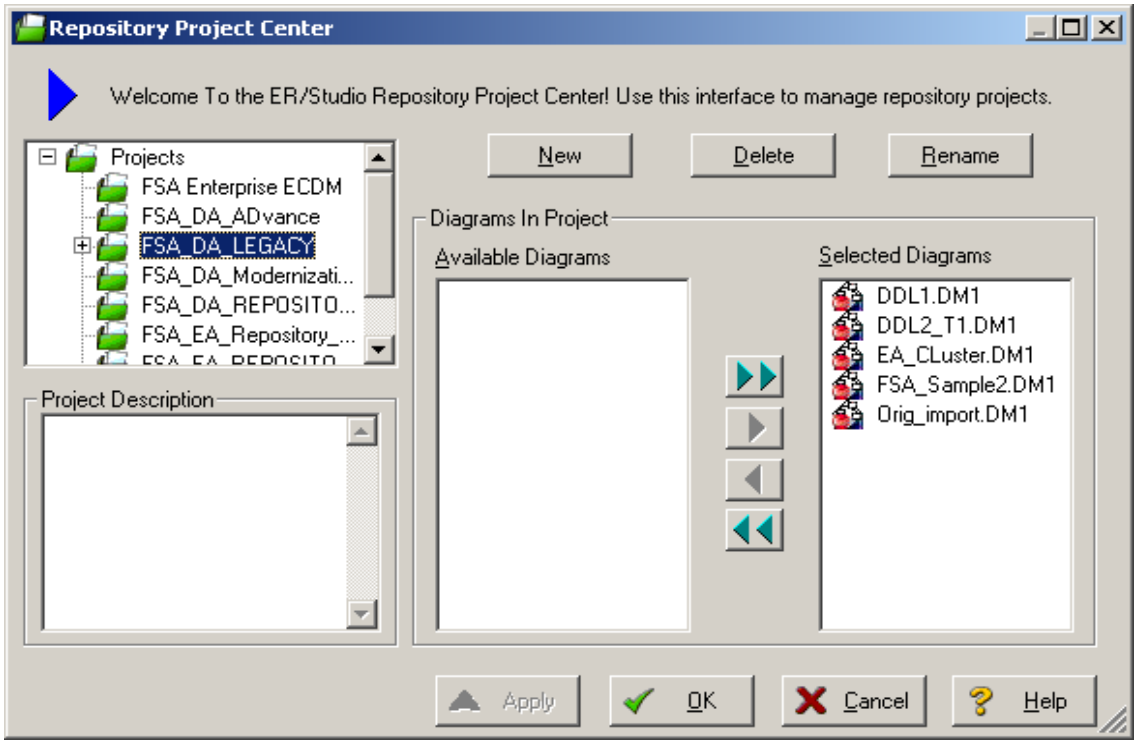


Figure L-9: Close-up of the Repository Project Center.

Under the "FSA_DA_LEGACY", create a new project called "Advance"

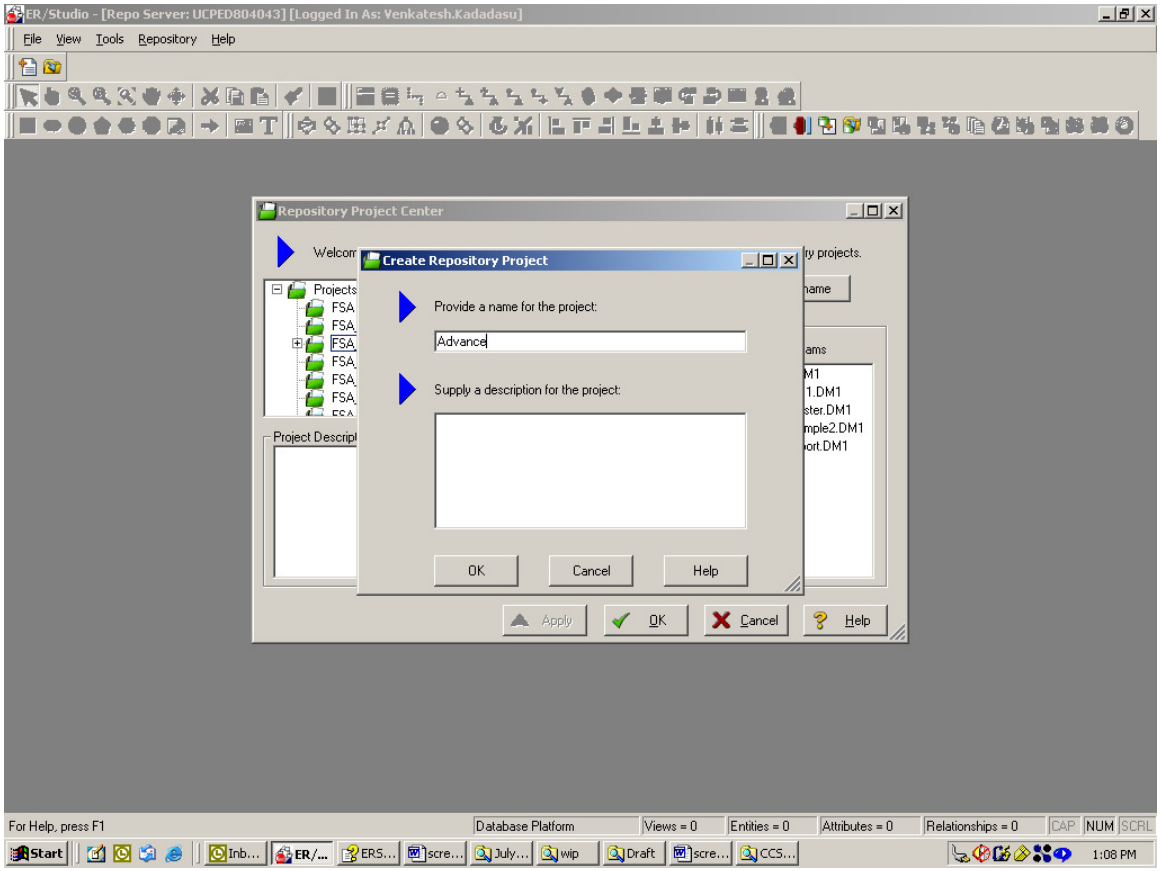


Figure L-10: Creating a project repository.

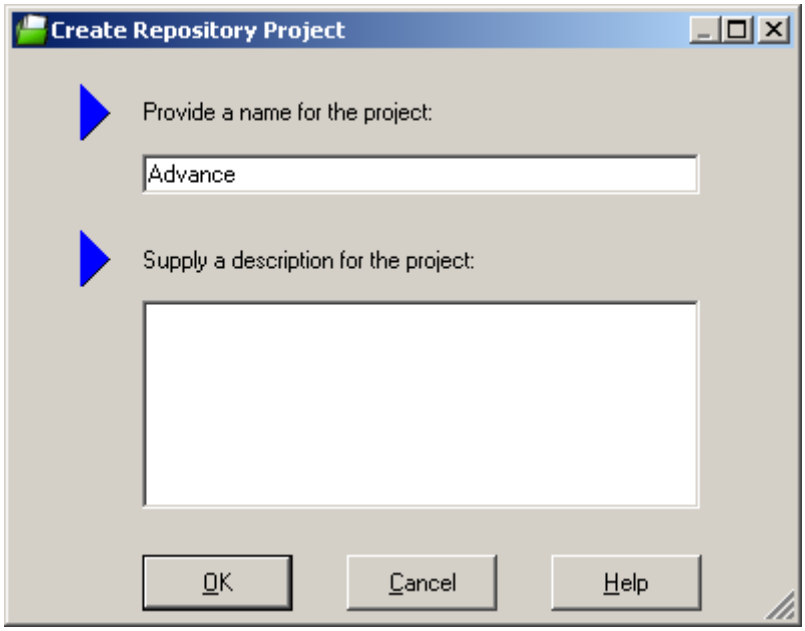
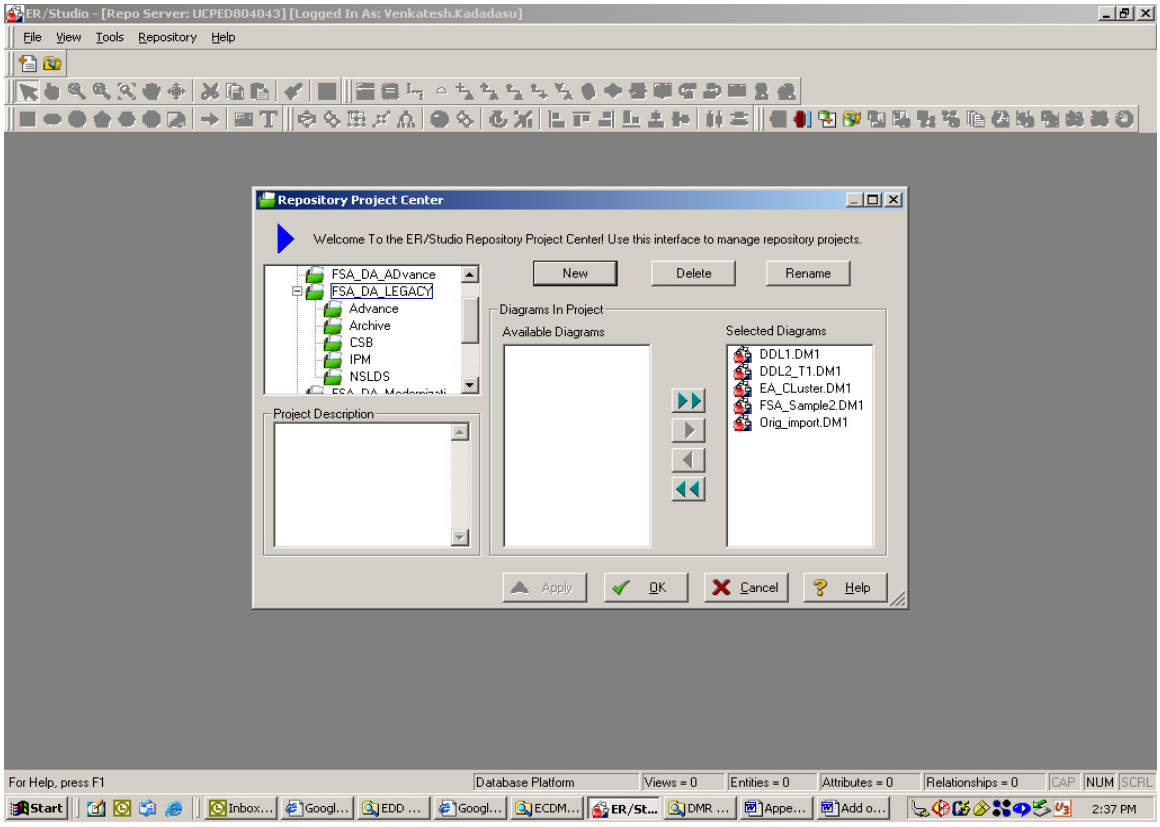
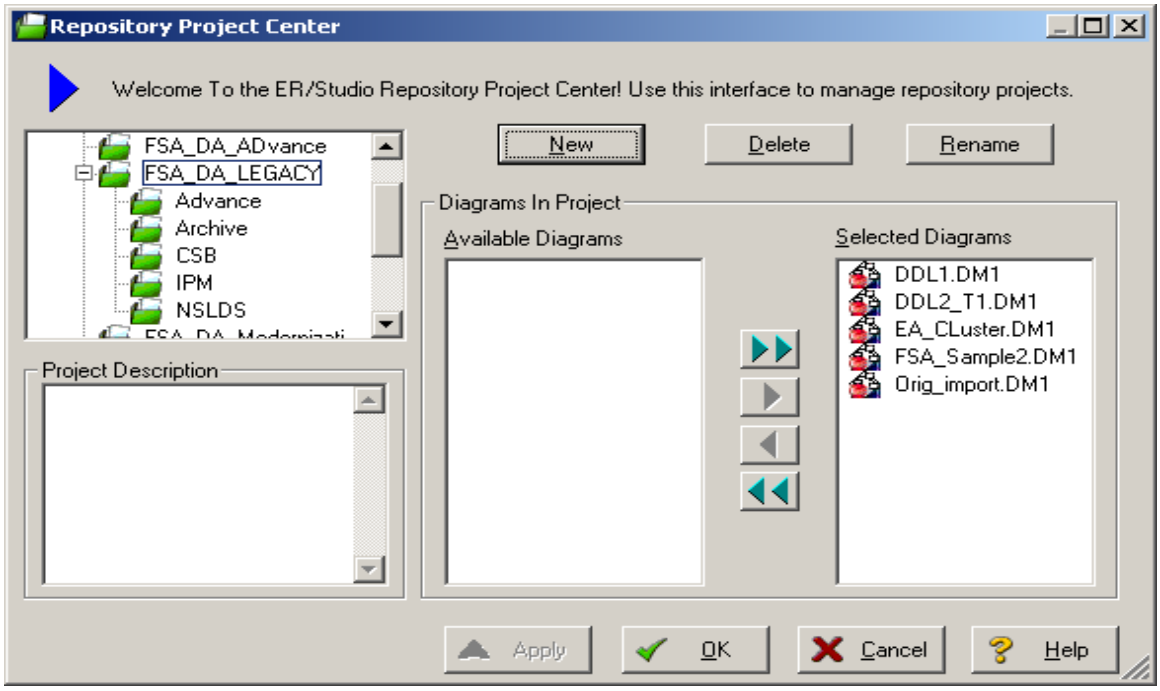


Figure L-11: Close-up of creating a project repository.





The Advance project repository folder now appears in the Project Center.

Repeat the above steps to set up the Project Folder Structure as defined in Section 3.1.

Adding Data Model Diagrams to ER/Studio Repository

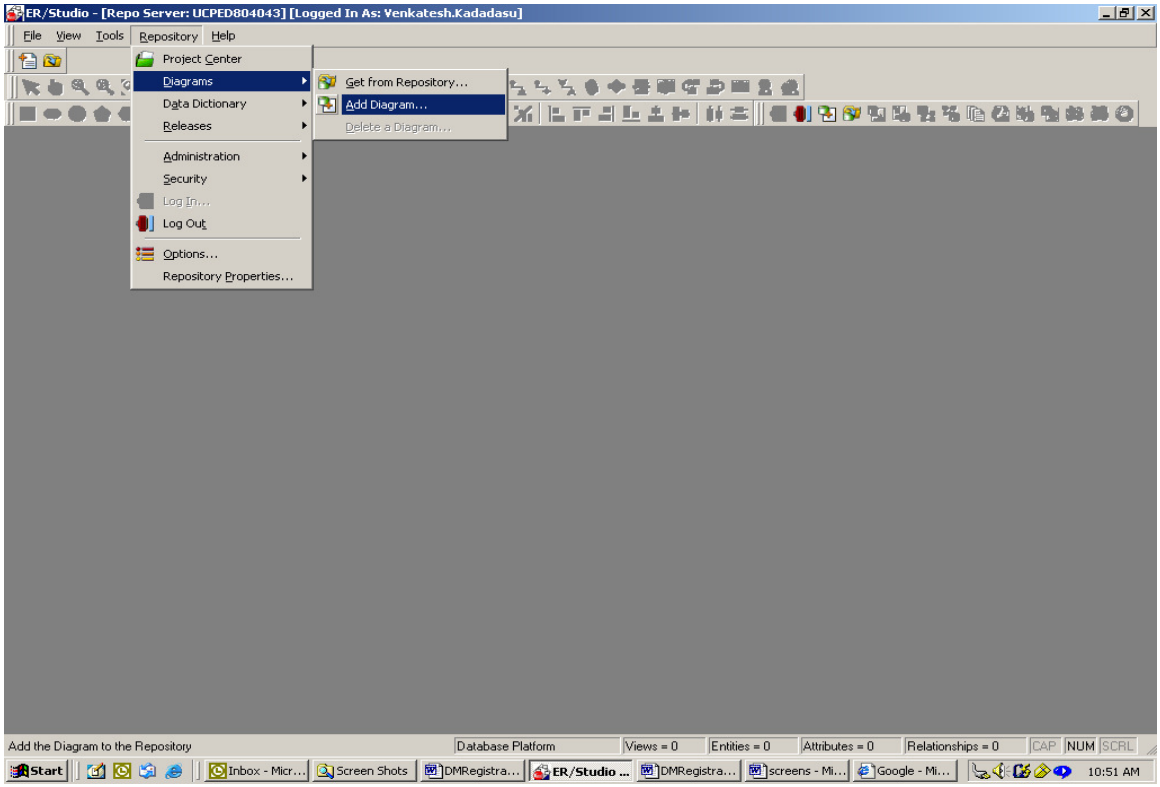


Figure L-12: Close-up of ER/Studio Repository Diagrams menu.

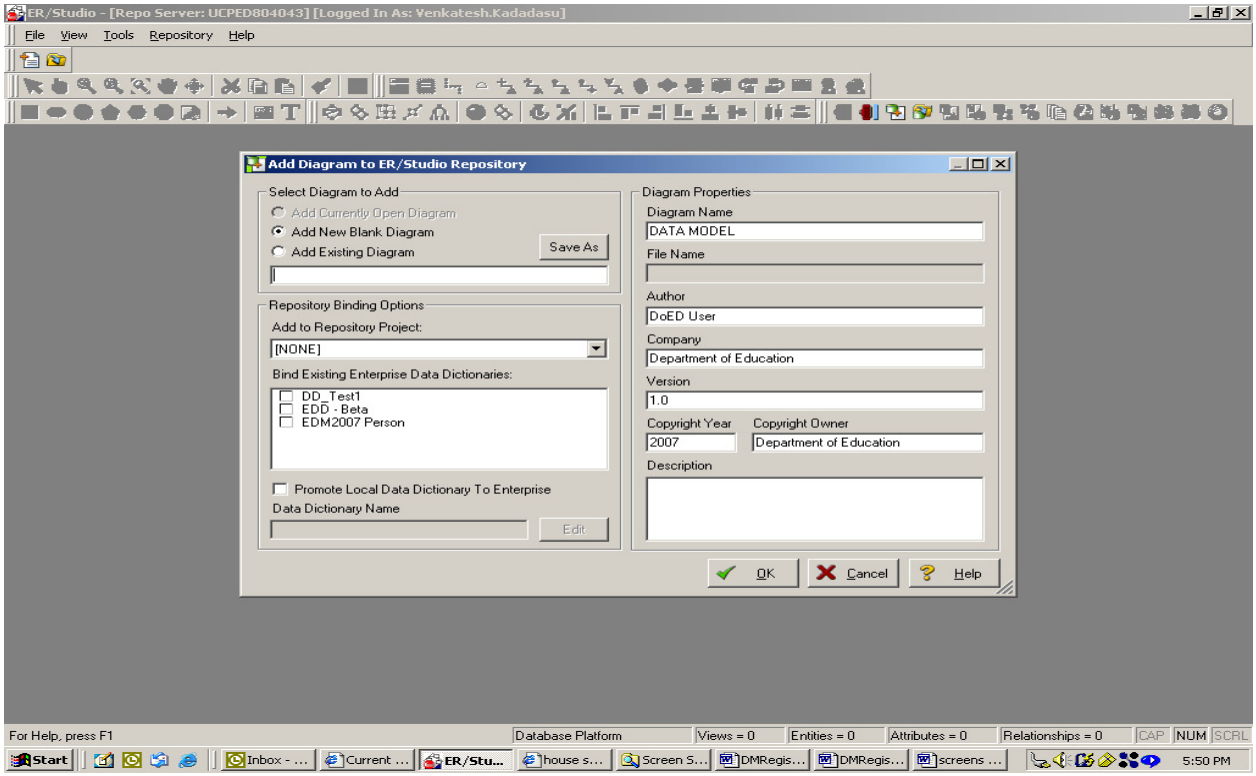


Figure L-13: Adding a diagram to the ER/Studio Repository.

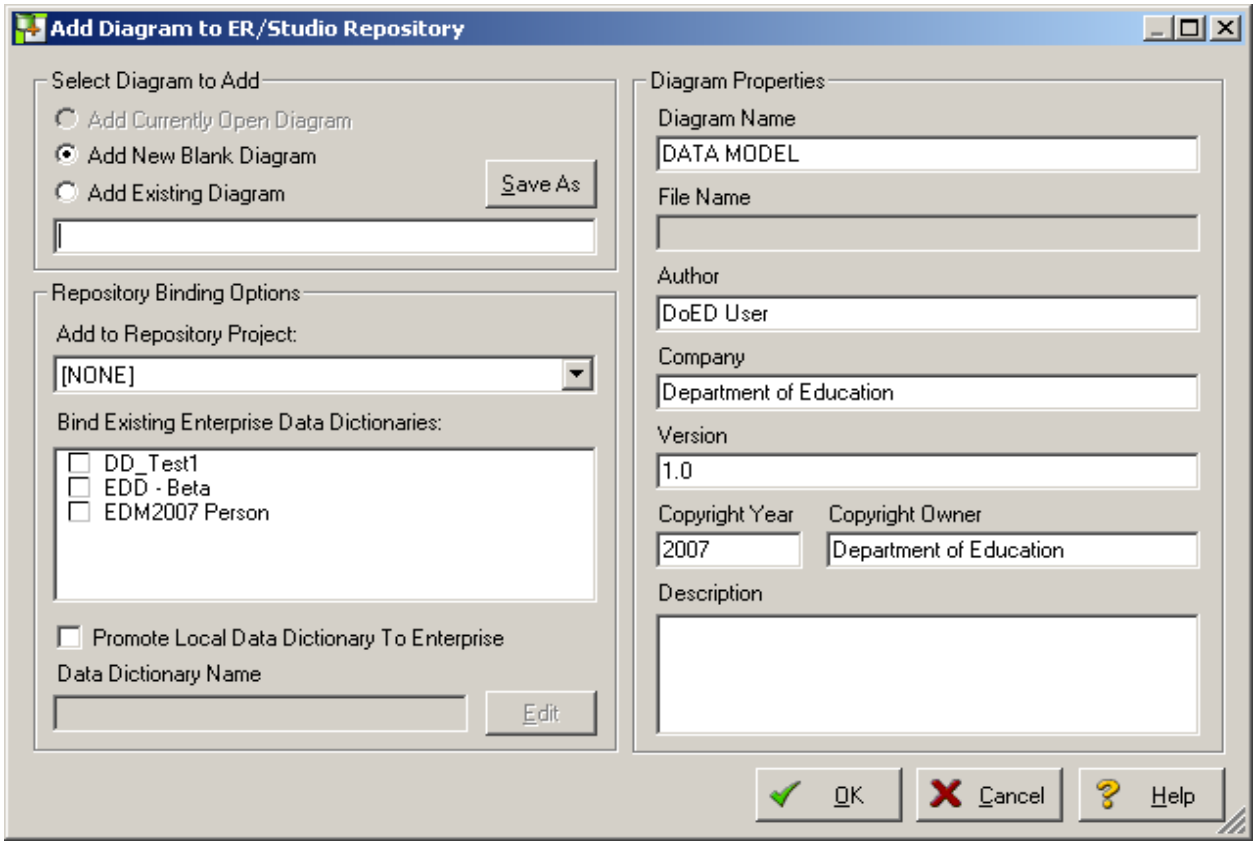


Figure L-14: Close-up of adding a diagram to the ER/Studio Repository

Appendix M. Converting Data Types

XML data XML data Type and Data Modeling Data types are different. The table below demonstrates the corresponding data types supporting data mapping and data synchronization efforts.

Embarcadero Data Type	Maps to DB2 Data Type	Comments	XML Types
PICTURE	BLOB (1k)	Specify length in the Definition	
BIGINT	DECIMAL (19,0)	Only for compatibility with UDB	
DECIMAL l s	DECIMAL (l,s)	greater than 999999999 or dollars	Decimal, Integer (when used for Amounts), Boolean, Integer (greater than 999999999)
DOUBLE PRECISION	DOUBLE PRECISION		
FLOAT	FLOAT		
INTEGER	INTEGER	up to 999999999	Integer (not greater than 999999999 and is not an amount), Token
REAL/SMALLFLOAT	REAL		
SMALLINT	SMALLINT	max value 4 digits	
CHAR n	CHAR (n)	fixed length char	Date (partial not full CCYYMMDD)
NCHAR n	GRAPHIC (n)		
NTEXT/LONG NVARCHAR	DBCLOB (1K)	Specify length in the Definition	
NVARCHAR n	VARGRAPHIC (n)	for say Japanese	
TEXT	CLOB (1K)	Specify length in the Definition	
VARCHAR n	VARCHAR (n)		String, String list
DATE	DATE		Date (full CCYYMMDD)
TIME/DATETIME	TIME		DateTime
TIMESTAMP/DATE	TIMESTAMP		

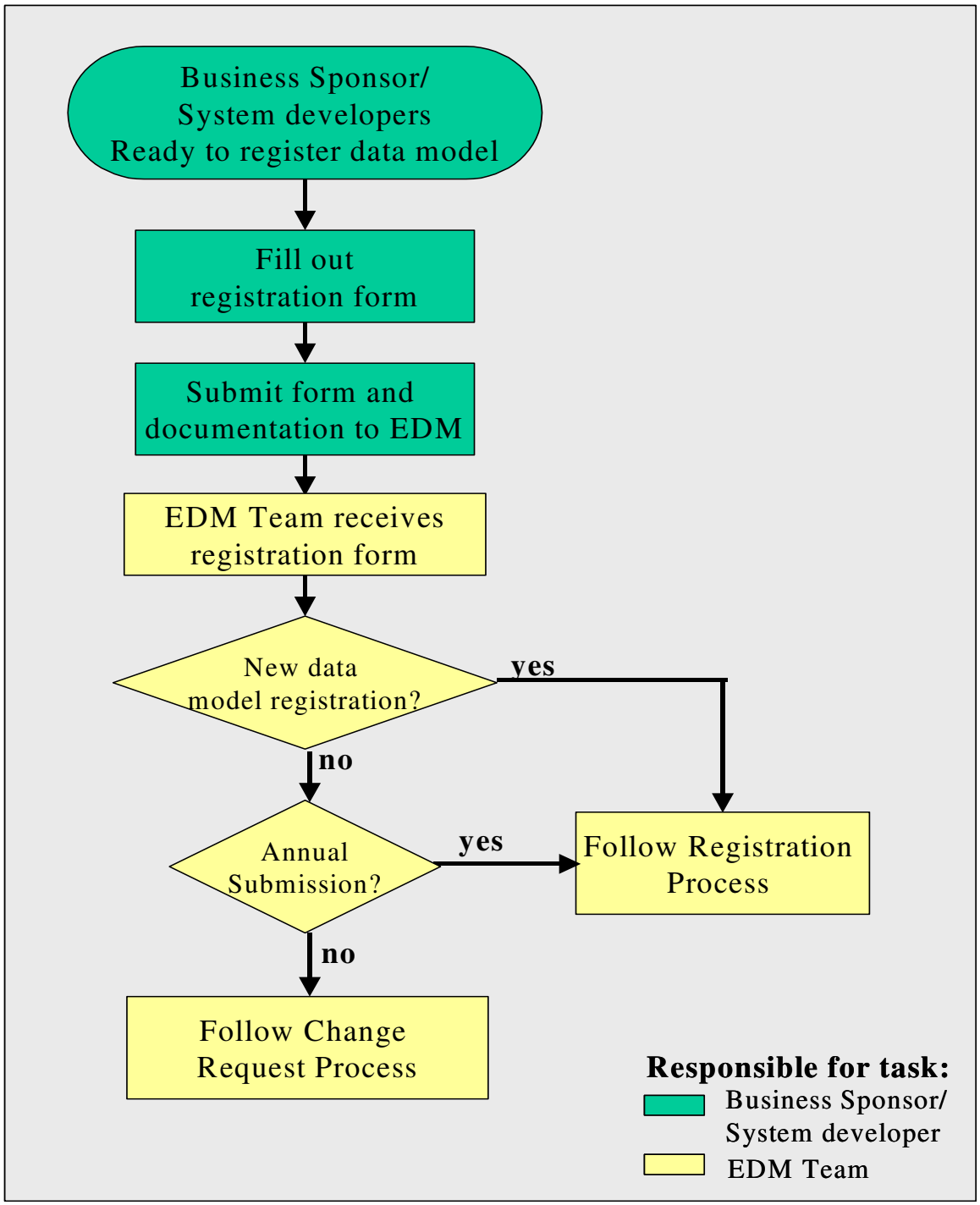
Appendix N. XML Crosswalk

The table below demonstrates how "components" in the XML Registry and Repository correlate to "components" of a data model.

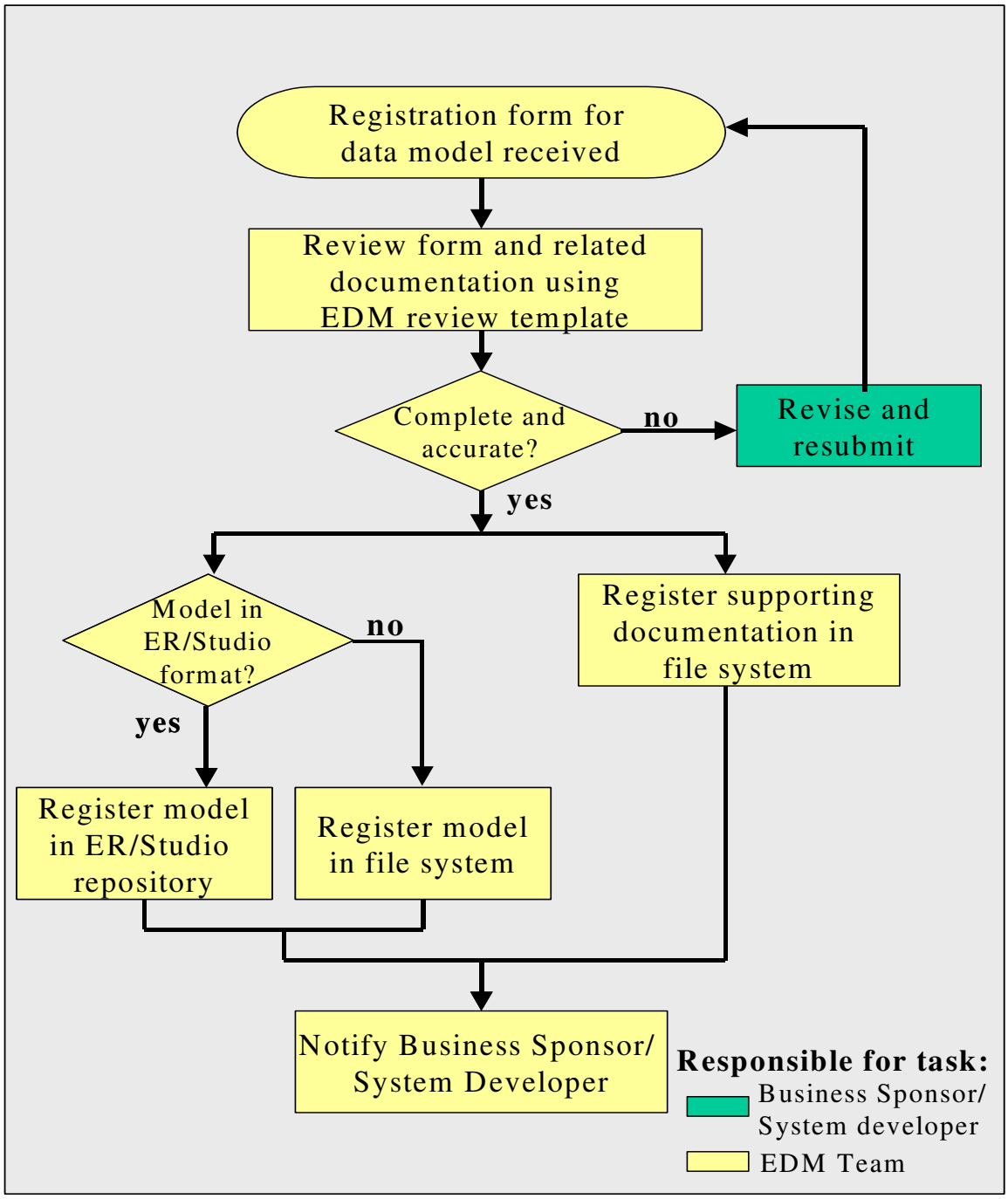
XML Registry	Logical Data Model
Parent Classification	Subject Area
Classification	Entity
Business Term (Simple Tag Type)	Attribute
Definition	Definition
XSD Base Type	Data Type
Enumeration List	Domain Values
Facet Value "maxLength"	Precision

Appendix O. Data model registration process and ECDM Change Request process flows

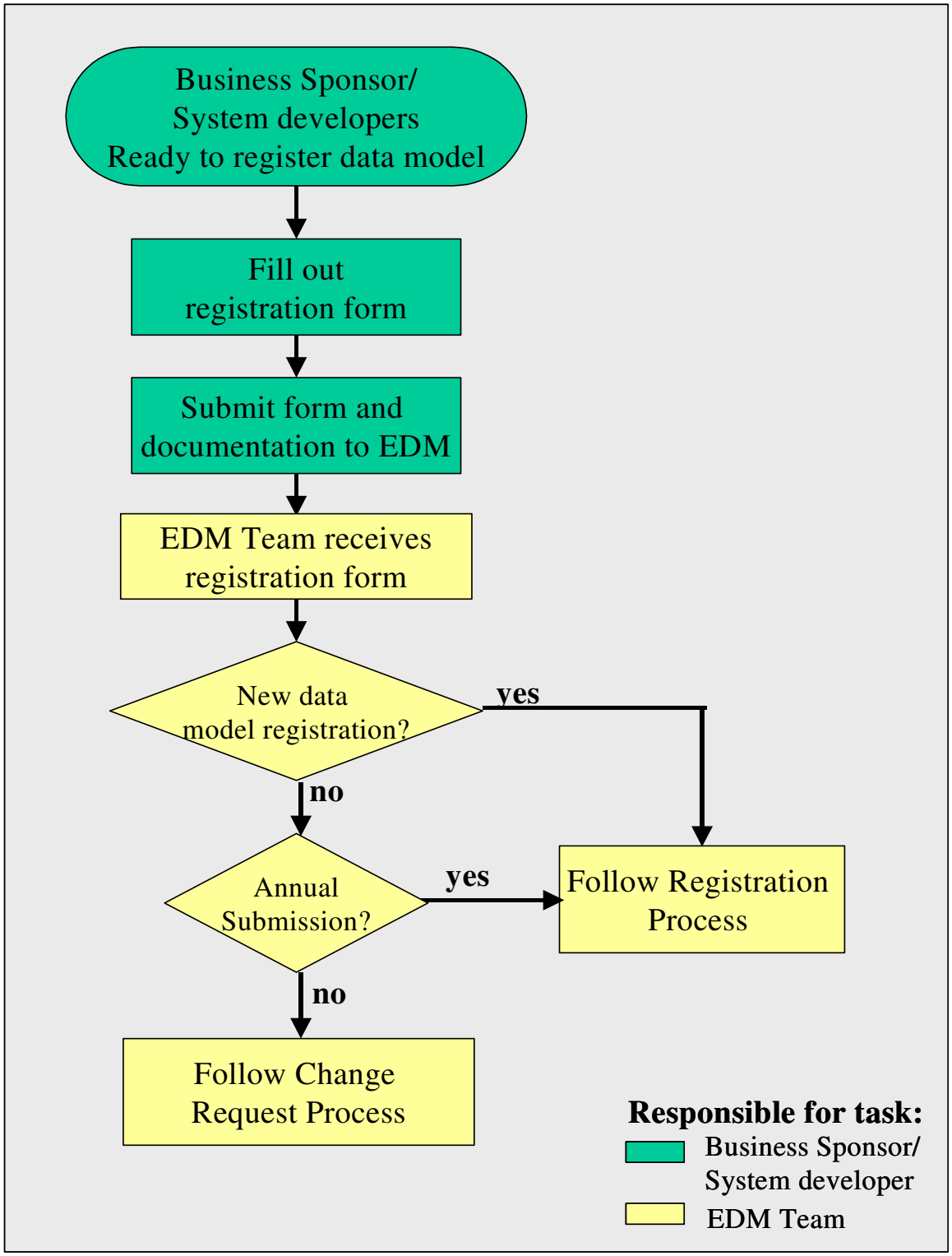
Data model submission by system developer to the EDM Team for review and registration.



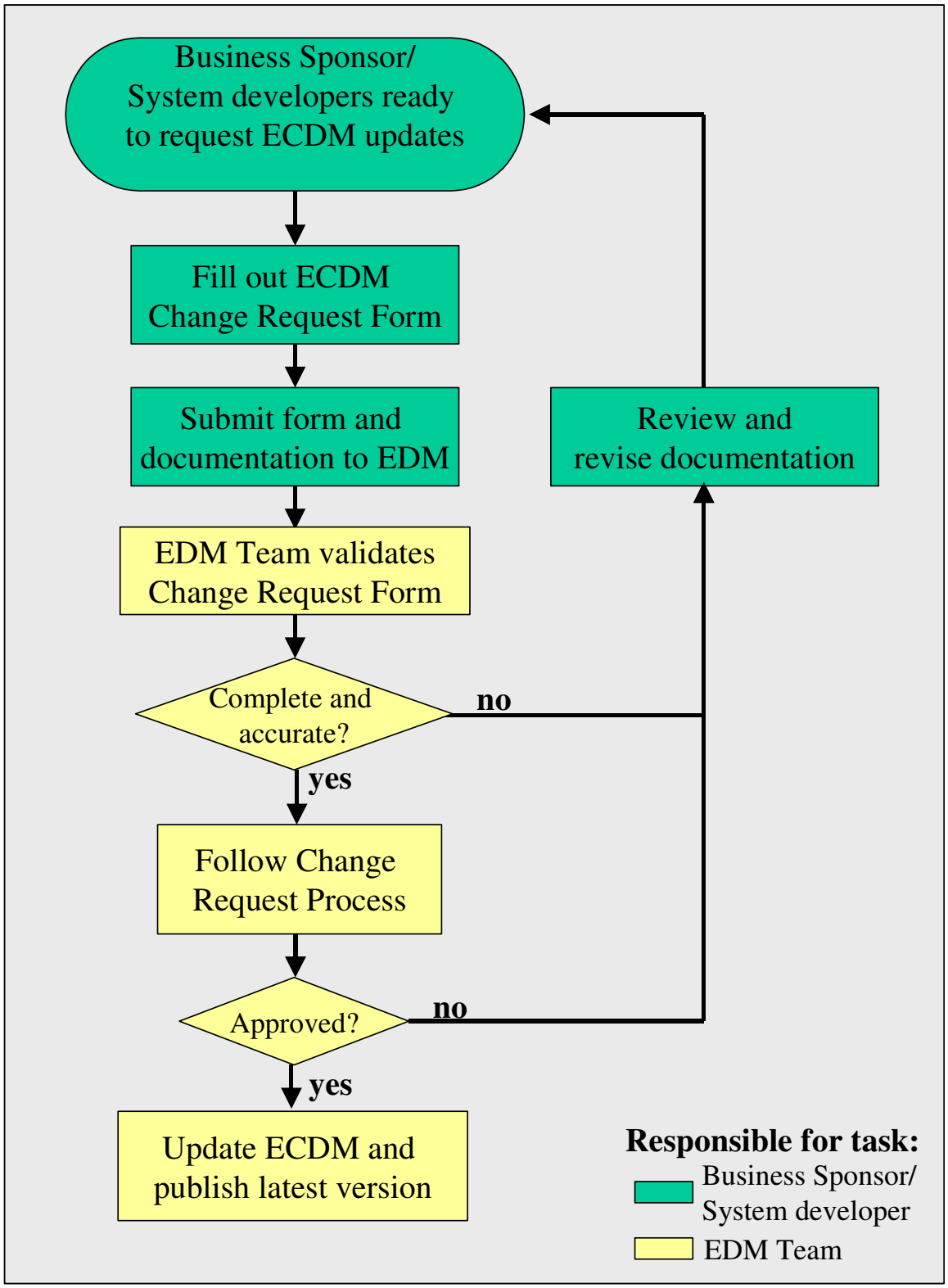
Data model registration process flow for any type of data model to be followed by the EDM Team upon receipt of the data model registration form.



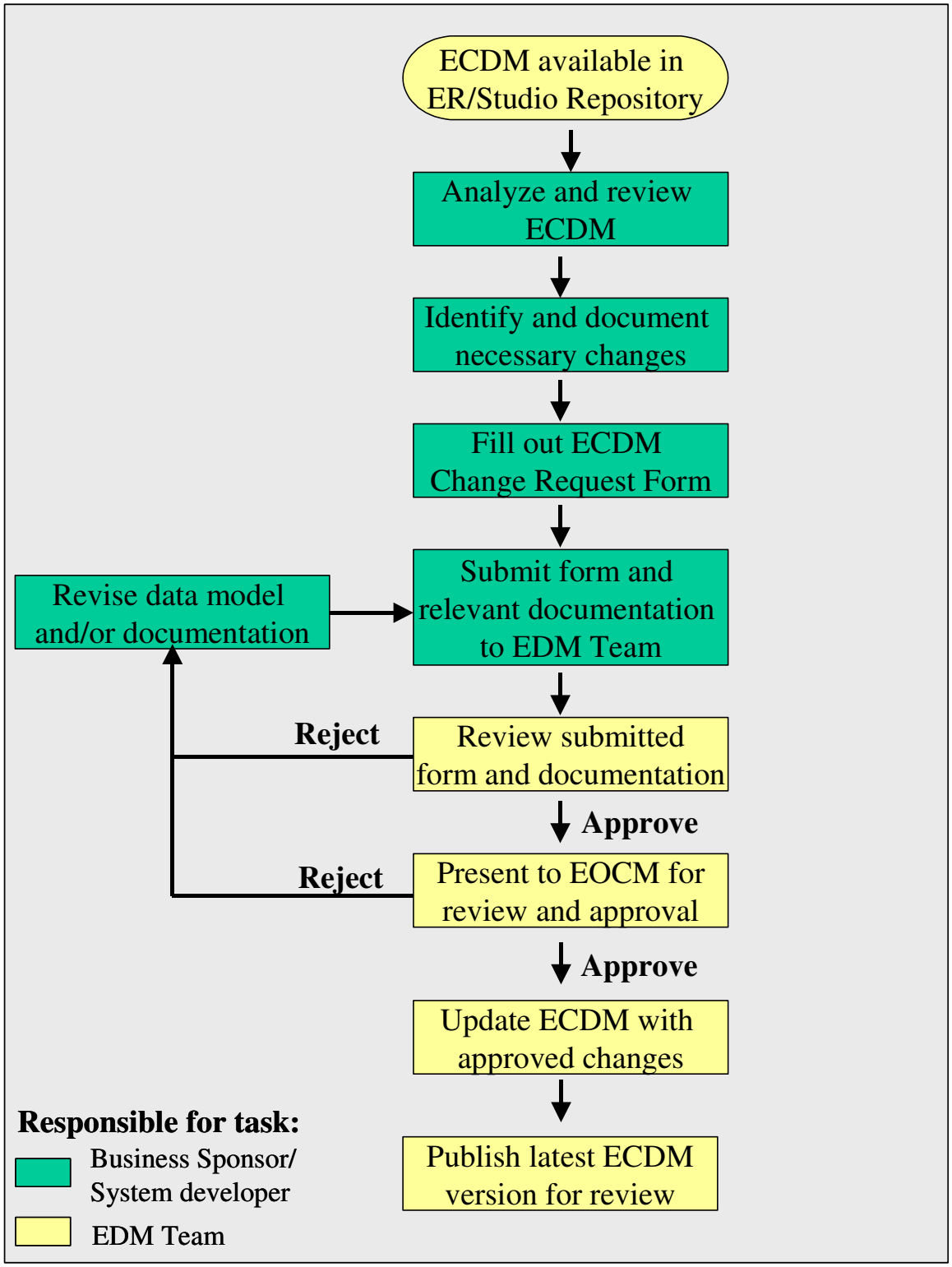
Data model resubmission process in case of modifications or annual submission.



ECDM Registration Process after identification of changes by business sponsor or system developers.



ECDM Change Request process flow as referenced in previous diagram.



Appendix P. Data model registration templates

Data Model registration template

To be completed by the System Developer	
1. Name of the organization or business unit submitting the data model	
2. Name of the Project	
3. Is the data model developed using ER/Studio? [Yes/ No]	
4. Is this data model submitted for first time registration? [Yes/ No]	
5. Reason for submission	
6. Is the submitted data model is Conceptual, Logical or Physical model?	
7. Specify the version of the current data model	
8. Name of the creator of this data model	
9. Business sponsor	
10. Date created	
11. Date submitted	
12. Additional information: (For Annual submission : provide brief summary of changes since last submission)	
Attachments:	

To be completed by the EDM Team	
1. Receipt date	
2. Registration Date	
2. Status of the request and Date [Approved / Rejected]	
3. Approved	
4. EOCM approval date	
5. Documents registered in the Repository? [Yes/ No]	

Conceptual/Logical data model registration template

The template consists of multiple worksheets: One worksheet for all entities and one worksheet for the attributes relating to each entity. The template also includes instructions on how to use it. Contact the EDM Team to obtain the template in electronic format.

Logical Data Model - Entity Info

	Business Unit Information	
	Business Unit Name	enter text here
	Data Model Name	enter text here
	File Name of the Data Model	enter text here

	Subject Area Information
	Subject Area Name
	enter text here

	Entity Information		
1	**Entity Name**	**Entity Type***	**Definition**
	enter text here	Independent	enter text here
		Dependent	

#	Relationship Information			
	Entity Name	**Parent/Child***	**Relationship Type***	**Cardinality***
1	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly

2	**Entity Name**	**Parent/Child***	**Relationship Type***	**Cardinality***
	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly

3	**Entity Name**	**Parent/Child***	**Relationship Type***	**Cardinality***
	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly

Logical Data Model - Attribute Info

	Business Unit Information	
	Business Unit Name	enter text here
	Data Model Name	enter text here
	File Name of the Data Model	enter text here
	Entity Information	
	Entity Name	enter text here

#	Attribute Information				
	Attribute Name	Data Type	Null*		Definition
1	enter text here	enter text here	a. null		
			b. not null		enter text here
2	Attribute Name	Data Type	Null*		Definition
	enter text here	enter text here	a. null		
			b. not null		enter text here
3	Attribute Name	Data Type	Null*		Definition
	enter text here	enter text here	a. null		
			b. not null		enter text here
4	Attribute Name	Data Type	Null*		Definition
	enter text here	enter text here	a. null		
			b. not null		enter text here

Physical data model registration template

The template consists of multiple worksheets: One worksheet for all tables and one worksheet for the columns relating to each table. The template also includes instructions on how to use it. Contact the EDM Team to obtain the template in electronic format.

Physical Data Model - Table Info

Business Unit Information	
Business Unit Name	enter text here
Data Model Name	enter text here
File Name of the Data Model	enter text here
Database Platform	enter text here

Subject Area Information
Subject Area Name
enter text here

Table Information		
Table Name	**Table Type***	**Definition**
enter text here	Independent	enter text here
	Dependent	

#	Relationship Information			
	Table Name	**Parent/Child***	**Relationship Type***	**Cardinality***
1	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly
2	**Table Name**	**Parent/Child***	**Relationship Type***	**Cardinality***
	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly
3	**Table Name**	**Parent/Child***	**Relationship Type***	**Cardinality***
	enter text here	Parent	Identifying	One to Zero or More
		Child	Non-Identifying	One to One or More
				One to Zero or One
				One to Exactly

Physical Data Model - Column Info

	Business Unit Information			
	Business Unit Name	enter text here		
	Data Model Name	enter text here		
	File Name of the Data Model	enter text here		
	Database Platform	enter text here		
	Table Name	enter text here		
#	**Column Information**			
	Column Name	**Data Type**	**Null***	**Definition**
	enter text here	enter text here	enter text here	enter text here
#	**Keys Information**			
	Column Name	**Type**	**Key Name**	**Restriction**
	enter text here	enter text here	enter text here	enter text here

Appendix Q. Data Modeling Essentials

Data modeling is an essential component when defining the requirements for a new system. It provides a blueprint for development. A good data model ensures that the resulting database will provide reliable, accurate and timely information. Yet despite it's importance data models are often overlooked by the business community. The fault does not necessarily reside with the users but is often the result of models that are overly complex, difficult to read and comprehend by both users and modelers alike. A good model adheres to standards regarding naming conventions and the manner in which the diagrams in the model are constructed.

Data Modeling is used to describe key business information and their relationships. The Entity Relationship (ER) model and its associated diagramming techniques is the method used to create the Relational Model and is the means of conveying a conceptual and logical data model. It presents an implementation-independent view of the data structures. A conceptual data model (CDM) is the means of collaboration between the data modeler and the business stakeholders and represents data in business terminology and provides high level information about the subject areas and data structures of an organization. The CDM is the key input for the logical design phase. The logical data model (LDM) builds on the CDM by developing all the attributes that will be required by the entities, and ensuring the completeness and integrity of the data. The LDM represents the business requirements and captures the required data and business rules of an organization. It creates structures that can be implemented into a database. The principal components for both the CDM and LDM are entities, attributes and relationships that roughly correspond to tables, columns and keys at the physical level. The LDM is the specification of the data to be held in the database. The physical data model (PDM) is the conversion of the LDM into a database structure. Its components are tables, columns and identifiers and may differ from the LDM due to performance considerations.

Naming and Definition Conventions

Naming and definition conventions are important when titling and defining entities, attributes and relationships. They should be of significance to the business community and readily understood. Cryptic naming obscures the meaning of what the purpose of these elements in the model and reinforces people's reluctance to review and use them. See section 2.3.1 and 2.3.2 in the "Data Model Standards and Guidelines, Registration Policies and Procedures" for further discussion of naming standards

Entity

An Entity is a categorization of things of interest to a business. They are recognizable concrete, tangible or abstract concepts such as persons, places things or events that are important to the business. They can be considered analogous to a table in a physical data model. ntities have a name and a definition. The name given to an Entity should be singular. For example use Grant instead of Grants, Person rather than Persons. Names should clearly define the object using terminology common to the business and should avoid using unnecessary abbreviations. Entities should avoid adding prefixes. For example, External Employee versus Employee. When creating a CDM prefixes can be used if they help clarify the model to the business community, but they may be stripped away during the logical data modeling phase.

Entity definitions should be clear, unambiguous, and succinct. They should distinguish the difference between entities as well as instances of an entity. They should provide guidance on the correct use of the information. For example, the following definition of the entity LOAN provides these characteristics:

Loans consist of Federal Direct Subsidized Loans, Federal Direct Unsubsidized Loans, Federal Direct PLUS Loans and Federal Direct Consolidation Loans. Schools award these loans, but students and parents repay them directly to the federal government.

Attribute

Attributes are information that you want to know about Entities. They are not captured in the CDM, but are developed and displayed in the LDM. For example, for the AID entity you may want to track the type of program offered (AID program) and the kinds of loan they tender (AID Loan Type). When naming attributes it is important to take a consistent approach in order to convey the meaning of each attribute as clearly as possible. If there are naming conventions established they should be adhered to. The attribute name should communicate the business concepts that users and business specialists are familiar with and that can be validated and verified. For example an attribute entitled "process" is unclear as to it's intent.

The same rigor should be applied when providing a definition for an attribute. A proper definition should impart what the attribute is intended to record. For example, for the attribute "Financial Aid Type" compare the following definitions:

- A kind of financial aid
- A form of financial aid that does not need to be repaid and that is ordinarily awarded on the basis of financial need. Sometimes used interchangeably with the term scholarship, though scholarships are generally awarded on the basis of merit.

The first definition is unclear in relation to how it interacts with the business, while the second provides background and information that can easily be validated by the user community.

A definition should provide clarity and not restate the obvious. Compare the following examples for the attribute "Student Identifier":

- The identifier for a Student
- The Student Identifier is the unique number assigned to a student on the date they apply for student aid. This number stays with the student throughout the aid process. It is a counter, which is incremented for each new student.

A review should be done to ensure that there are no attributes with the same or similar names that have different definitions or attributes with the same name but different definitions.

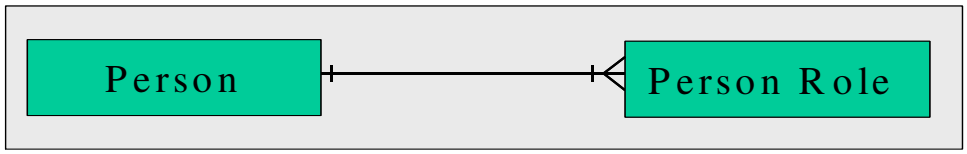
Relationship

Data models also show relationships between entities. For example there is a relationship between Student and Loan. Relationships are notated to describe and help clarify the association: A Student applies for a Loan. Notations should accurately reflect and describe the relationship between entities. The relationships also define the constraints between entities by diagramming

whether an association is optional and whether there is a limitation on the number of instances that can occur between the entities. For example "A Student can apply for zero or more Loans, and a Loan is issued to only one Student." This statement tells us that a student may choose not to apply for a loan but also that they can apply for several loans. It is imperative that the constraints be properly defined as they help enforce the business rules.

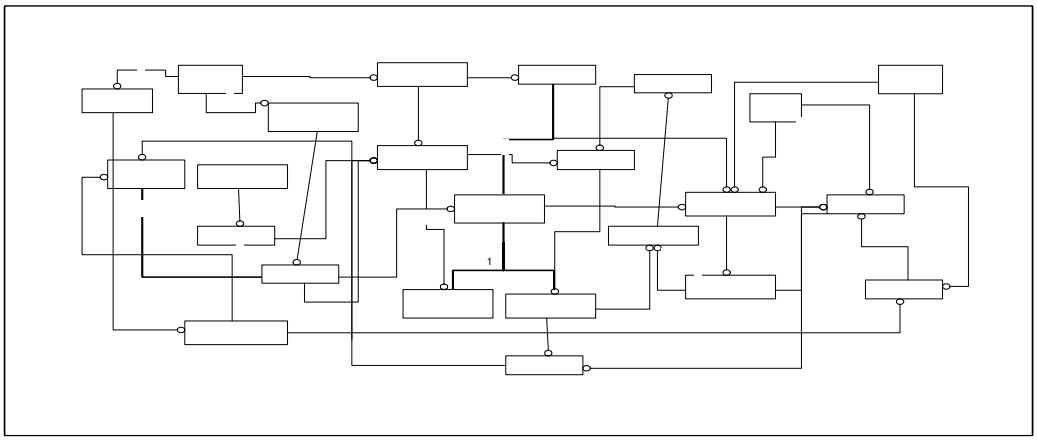
Aesthetics

An Entity Relationship diagram (ERD) displays entities within a box and shows the relationships, including cardinality through lines arrows:



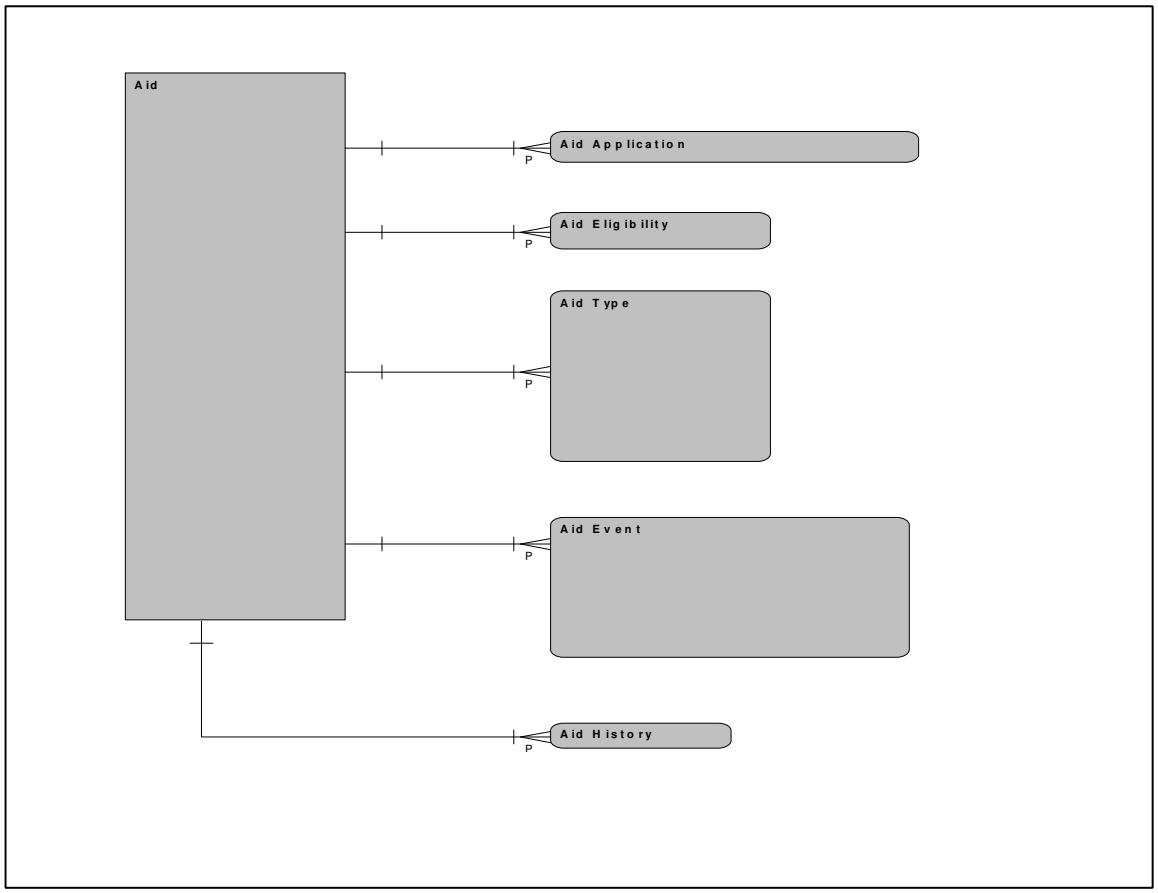
The diagram above shows two entities, Person and Person Role. The line connecting the two entities shows there is a relationship between them. The "crows feet" at the end of the line at Person Role indicates that Person is the parent entity and that Person Role is the child entity. It also indicates that a Person can assume one or more roles. More details on how to read the relationship between these two entities are captured in the Communication section at the end of this document.

Models, specifically ERDs, can become complex and indecipherable if the modeler does not account for the visual effect it will have for an user trying to navigate through it. Models become burdensome when a one tries to do too much with them. There is a little to the amount of information that can be displayed. A diagram with numerous entities and multiple relationships can become a maze of "boxes and lines" if not designed properly.



The simplest thing to do to improve the appearance of a model is to limit the number of entities displayed. A diagram should display no more than 20 entities, but 10 to 15 are preferable. For large models, views or subject areas can be created to accommodate this. How entities are placed in a model can affect the viewer's ability to understand a diagram. Some people arrange entities to minimize having relationship lines cross with no particular business purpose in mind.

While this improves the readability, it may not impart value in understanding the overall meaning, as the viewer may still observe a random collection of "lines and boxes". It may not distinguish an independent entity (an entity that that does not rely on another entity for identification and reflect tangible things that are important to a business) from a dependant entity (an entity that relies on another entity for identification and represent transactional information that represent what the business does). A convention for overcoming this obstacle is to display relationship lines that point in the same direction. This separates reference entities that describe tangible things (e.g. PERSON, AID, ORGANIZATION) in the business from entities describing what the entity does (AID APPLICATION, AID ELIBILITTY). By having the "crows feet" point left an up, we can readily discern the substantial entities from the reference entities.



The diagram above clearly shows that AID is the tangible thing that is important to the business and the other entities represent the transactions the business conducts.

Benefits of a good data model

By following the guidelines listed above when developing entities, attributes and relationships the modeler will be able to create a data model that will provide utility to the business community. When translating the CDM into a LDM these guidelines should continue to be adhered but other criteria must be evaluated to ensure the quality of the logical data model and it's capability to support the business requirements. LDMs should be normalized to ensure that there is consistency and quality in the data. Normalization is the process of efficiently

organizing data in a data model or database. Additional information on data model/database normalization is available in the Communication section at the end of this document.

Completeness

An LDM ensures that all relevant and necessary data is captured and represented accurately. For example, if the model lacks the means to record loan payments and this data is required by the system this would be a serious omission.

Non-Redundancy

A LDM ensures that information is only recorded once. This is done through a technique known as normalization. Several legacy systems record facts in more than one table and sometimes in several systems. Capturing data in this fashion essentially duplicates it and could result in anomalies. If a data item is stored in more than one place we cannot be confident it is always represented consistently. There are also maintenance issues. If the data item needs to be updated can we confirm that it has been updated in each table?

Enforcement of Business Rules

The LDM must reflect and enforce the rules that apply to the business data. If the model enforced that a Student could apply for only one loan then this would not support the business requirement. If the rule correctly enforces the business requirement(s) in the logical data model the resulting database will implement and enforce the correct practices and maintain the quality of the data.

Data Reusability

The LDM should be constructed independent of an application or a system. Information captured by one organization or department may want to be used by other groups. A Loan Officer could later utilize demographic information for a Student captured during an inquiry about Aid. Structuring the data independent off either application allows both processes to employ it.

Stability and Flexibility

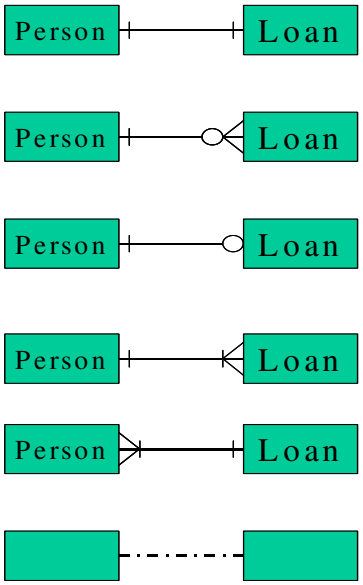
An LDM should be able to support changes to the business requirement. If new data is required the model should be able to accommodate it into existing tables or with minor extensions to the model.

Communication

Conceptual and logical data models should represent the business concepts that the users and business specialists are familiar with. Business owners and stakeholders should verify that it is accurate. Conceptual and logical data models can be organized into subject areas that reflect different parts of the organization to assist in this endeavor. Logical data models need to be presented at different levels of detail to allow the reviewer the capability to grasp what has been captured.

Components of an ER-Diagram

Symbol	Description
Person	Entity
———	Relationship
	Cardinalities:
|	Exactly one
○	Zero (optional)
< >	More than one (multiple)



One Person has exactly one Loan;
One Loan refers to exactly one Person.

One Person has zero or many Loans;
Zero or many Loans refer to exactly one Person.

One Person has zero or one Loan;
One Loan refers to exactly one Person.

One Person has one or many Loans;
One or many Loans refer to exactly one Person.

One Person has exactly one Loan;
One Loan refers one or many Persons.

Dotted line indicates relationship to "context entity"

Data Normalization

a) The Technique

Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types of logical or structural problems. For example, when multiple instances of a given piece of information occur in a table, the possibility exists that these instances will not be kept consistent when the data within the table is updated, leading to a loss of data integrity. A table that is sufficiently normalized is less vulnerable to problems of this kind, because its structure reflects the basic assumptions for when multiple instances of the same information should be represented by a single instance only.

Higher degrees of normalization typically involve more tables and create the need for a larger number of joins, which can reduce performance. Accordingly, more highly normalized tables are typically used in database applications involving many isolated transactions (e.g. an automatic teller system), while less normalized tables tend to be used in database applications that do not need to map complex relationships between data entities and data attributes (e.g. a reporting application, or a full-text search application).

Database theory describes a table's degree of normalization in terms of normal forms of successively higher degrees of strictness. A table in third normal form (**3NF**), for example, is consequently in second normal form (**2NF**) as well; but the reverse is not always the case.

Although the normal forms are often defined informally in terms of the characteristics of tables, rigorous definitions of the normal forms are concerned with the characteristics of mathematical constructs known as relations. Whenever information is represented relationally, it is meaningful to consider the extent to which the representation is normalized.

b) Rules of Data Normalization

As mentioned above, there are several levels of normalization. A brief description is provided in the table below.[7]

Level	Description
1NF	Eliminate Repeating Groups - Make a separate table for each set of related attributes, and give each table a primary key.
2NF	Eliminate Redundant Data - If an attribute depends on only part of a multi-valued key, remove it to a separate table.
3NF	Eliminate Columns Not Dependent On Key - If attributes do not contribute to a description of the key, remove them to a separate table.
BCNF	Boyce-Codd Normal Form - If there are non-trivial dependencies between candidate key

[7] Source: http://www.datamodel.org/NormalizationRules.html

Level	Description
	attributes, separate them out into distinct tables.
4NF	Isolate Independent Multiple Relationships - No table may contain two or more 1:n or n:m relationships that are not directly related.
5NF	Isolate Semantically Related Multiple Relationships - There may be practical constrains on information that justify separating logically related many-to-many relationships.

Appendix R. Defining the Data - Constructing a well-written data element definition

The goal of the Enterprise Data Management program is to consistently define data and make standardized data available across the enterprise. According to the International Organization for Standardization (ISO)[8] "*Data processing and electronic data interchange rely heavily on accurate, reliable, controllable and verifiable data recorded in databases. A prerequisite for correct and proper use and interpretation of data is that both users and owners of data have a common understanding of the meaning and representation of the data.*" This common (and agreed) meaning and representation of the data is documented and published in the form of a data definition. The goal (and challenge) of creating a well-written definition is to ensure that the definition is specific enough to meet the needs of the organization, yet generic enough to be used across a community.

A well-written definition should explicitly describe and explain the meaning of the business term or data element. As the definition provides the context for which business is being conducted, each data element definition should consist of certain components and characteristics.

Definition Structure

Meaning nuances often occur based upon context, it is important to clearly describe the concept/term, providing as much information as possible to limit such occurrences. A well-written definition should incorporate at least 2 of the following components[9] as part of the definition text:

- Broader Term – a general class to which a concept/term belongs, often this is implied. To better explain this concept, consider an "IS A" relationship. For example, "A school *is a*n organization".
- Distinguishing Characteristics – the pertinent attributes with specific values of the term. To better explain this concept, consider a "HAS A" relationship. For example, "A school *has a*n academic program".
- Function Qualifier – how the concept/term being defined is used, usually involves verbs. To better explain this concept, consider a "USED FOR" relationship. For example, "A school is *used for* educating students".

An example of a good definition using the above components:

> *"A school is a learning organization that has one or more academic programs used for educating students"*
>
> Broader Term: *learning organization*
> Distinguishing Characteristic: *one or more academic programs*
> Function Qualifier: *used for educating*

[8] International Organization for Standardization. Information technology — Metadata registries (MDR) — Part 4: Formulation of data definitions (ISO/IEC 11179-4, Second Edition 2004) 4/18/2007.

[9] O'Neil, Bonnie "Business Metadata: How to Write Definitions". *Business Intelligence Network*. (March 29, 2005). 3/23/2007 http://ww.b-eye-network.com/view/734

In the creation of a data dictionary or glossary of terms, additional information may be incorporated into the entry for clarity. The following components[10] may be used (creating a data dictionary entry is beyond the scope of discussion here, therefore detailed explanation will not be provided herein. However, for information regarding the creation of a data dictionary or glossary, please refer to Federal Student Aid's Enterprise Data Dictionary Standards.):

- Name – the name of the term being defined
  - Narrower Term – the classes beneath/within (or belonging to) the term being defined
  - Related Term – a term that has relevance to the concept/term being defined but not a synonym
  - Synonyms – terms that mean nearly the same as the concept/term being defined.
  - Examples and usage – an instance of the concept/term as it is seen in everyday life.
  - Source – where the definition of the concept/term came from (originated).
  - Replaced By – a newer term, that has a similar meaning as the concept/term being defined, which would indicate the term being defined is not in common usage. This can also be noted as a Synonym but may be helpful in the case of migration or reverse engineering.
  - Approval – information about when the concept/term definition was approved, by whom, etc.

Definition Requirements

Understanding that the context in which the data is used is a key factor in defining data elements, generally, when composing a data element definition, each definition should have the following characteristics:[11]

- Unique - A definition should be unique and distinguishable from every other data element definition.
  EXAMPLE: "Closure Date"
  Poor Definition: The date when something closed.
  Good Definition: The date that the school ceases to provide educational instruction in all programs, as determined by the U.S. Secretary of Education.

- Clear – A data definition should be precise, concise, and unambiguous. The definition should be clear enough to allow only one possible interpretation.
  EXAMPLE: "Disbursement Date"
  Poor Definition: The date money was disbursed.
  Good Definition: The date money was credited to the student's account at the school or paid to the student (or borrower if a PLUS loan) directly.

- Singular - The data definition should be expressed in the singular.
  EXAMPLE: "OPEID"

[10] O'Neil, Bonnie "Business Metadata: How to Write Definitions".
[11] International Organization for Standardization. ISO/IEC 11179-4, Second Edition 2004. 4/18/2007

Poor Definition: The identification number assigned by OPE for data exchange partners.
Good Definition: The unique identifier assigned by the Office of Postsecondary Education (OPE) for each data exchange partner.

- Positive - The definition should be expressed as what it is, limit any emphasis on what it is not.
  EXAMPLE: "Loan Net Amount"
  Poor Definition: The amount that doesn't include any fees.
  Good Definition: The total amount disbursed to the borrower after any fees or charges have been deducted.

- Specific Concept - The definition should include the essential meaning or primary characteristics of the concept.
  EXAMPLE: "Request Tracking ID"
  Poor Definition: The unique ID assigned to a document match and tracking request.
  Good Definition: The unique ID associated with a request action that is returned to the requestor for document matching and tracking.

- Defined with Commonly Understood Abbreviations - The definition should only use abbreviations when necessary and the abbreviation must be commonly understood.
  EXAMPLE: "EFC Available Income"
  Poor Definition: The available income used in the EFC calculation.
  Good Definition: The income available for education expenses used for determining financial need based on the Expected Family Contribution (EFC).

- Primary Definition - The definition should not contain any embedded definitions or underlying concepts of other data elements/terms/concepts.
  EXAMPLE: "EFC Available Income"
  Poor Definition: The available income used for determining the EFC, where the EFC (expected family contribution) is calculated by subtracting the cost of attendance from the expected family contribution.
  Good Definition: The income available for education expenses used for determining financial need based on the Expected Family Contribution (EFC).

- Expressed without rationale, functional usage, domain information or procedural information – The definition should not include statements about why and how a data element is used.
  EXAMPLE: "Disability Condition Status Code"
  Poor Definition: The codes used to identify the status of a disability condition.
  Good Definition: A code that identifies the confirmed status of a disability condition on a student aid record

- Defined without circular reasoning – The definition should not be defined in terms of another data element.

EXAMPLE: "Deferment"
Poor Definition: Deferment is a subtype of Aid Transaction
Good Definition: The temporary postponement of payments on a loan or debt.

An example of a good definition using the above components and requirements:

*"**Forced Collection**: An involuntary collection activity (Treasury Offset Program, Administrative Wage Garnishment, Litigation) performed by Federal Student Aid to collect repayment of a defaulted loan after all other methods, efforts, and attempts to collect have been exhausted.*

<u>Components Usage</u>
Name: *Forced Collection*
Broader Term: *involuntary collection activity*
Function Qualifier: *to collect repayment of a defaulted loan*

<u>Optional Components Usage:</u>
Narrower Term: *Treasury Offset Program, Administrative Wage Garnishment, Litigation*

<u>Requirements Usage</u>
Singular: *"activity"* refers to a single instance of forced collection
Positive: *"to collect"* refers to what forced collection is intended for as opposed to what it's not intended for.
Specific Concept: *"involuntary collection activity to collect repayment of a defaulted loan"* is the primary concept of *Forced collection*.