



Requirements Modeling and Specification Guidelines and Standards

[Based on the Use of Oracle Designer Repository]

Version 3. 2

March 31, 2003

Information and Technology Management Branch

❖ IM / IT Standards & Guidelines ❖

| | |
|--|-----------|
| 1. INTRODUCTION | 1 |
| 1.1. Purpose of this document | 1 |
| 1.2. Audience | 1 |
| 1.3. References | 1 |
| 1.3.1. IMG Documents | 1 |
| 1.3.2. Books and web resources | 1 |
| 1.4. Change History | 2 |
| 2. BACKGROUND | 4 |
| 2.1. What is modeling? | 4 |
| 2.2. Scope of modeling discussed in this document | 5 |
| 2.3. Information Management Context | 7 |
| 2.4. Fundamental Principles of Requirements Modeling | 8 |
| 2.5. Approach to Standards in the context of Oracle Designer | 9 |
| 2.6. Operational Infrastructure | 10 |
| 2.6.1. Requirements Modeling Tools | 10 |
| 2.6.2. Central Repository | 10 |
| 2.6.3. Repository Management | 11 |
| 2.6.4. Generalized Application Management Environment (GAME) | 11 |
| 3. DATA MODELING STANDARDS | 12 |
| 3.1. General Concepts | 12 |
| 3.2. Use of Enterprise Data Models | 12 |
| 3.3. Data Architecture Strategies | 13 |
| 3.4. Normalization | 14 |
| 3.5. Generalization and Abstraction | 15 |
| 3.5.1. Supertypes and Subtypes | 15 |
| 3.5.2. Representation of Types | 15 |
| 3.5.3. Domains | 16 |
| 3.5.4. Generalization of Attributes | 17 |
| 3.5.5. Rule representation | 17 |
| 3.6. Diagramming Guidelines | 17 |
| 3.6.1. Label | 17 |
| 3.6.2. Relationship Lines | 17 |
| 3.6.3. Colour | 17 |
| 4. ENTITY DEFINITION STANDARDS | 19 |
| 4.1. Entity Name | 19 |
| 4.2. Entity Short Name | 20 |
| 4.3. Plural | 20 |

| | | |
|-----------|--|-----------|
| 4.4. | Entity Volume Information | 20 |
| 4.5. | Synonym | 20 |
| 4.6. | Entity Description | 20 |
| 4.7. | Entity Notes | 21 |
| 4.8. | Time dependency | 22 |
| 4.9. | Unique identifier | 23 |
| 4.10. | Enterprise Level considerations | 23 |
| 4.11. | Initial Data Model considerations | 24 |
| 5. | RELATIONSHIP DEFINITION STANDARDS | 25 |
| 5.1. | Relationship Name | 25 |
| 5.2. | Optionality | 25 |
| 5.3. | Cardinality | 25 |
| 5.4. | Transferability | 25 |
| 5.5. | Exclusive Arc | 25 |
| 5.6. | Relationships among subtypes | 26 |
| 6. | DOMAIN DEFINITION STANDARDS | 27 |
| 6.1. | Name | 27 |
| 6.2. | Subset Of | 27 |
| 6.3. | Comment | 27 |
| 6.4. | Attributes in Domain | 28 |
| 6.4.1. | Format | 28 |
| 6.4.2. | Average Attribute Length | 28 |
| 6.4.3. | Maximum Attribute Length | 28 |
| 6.4.4. | Attribute Decimal Places | 28 |
| 6.5. | Unit Of Measure | 28 |
| 6.6. | Columns in Domain | 28 |
| 6.6.1. | Data type | 29 |
| 6.6.2. | Average Column Length | 29 |
| 6.6.3. | Maximum Column Length | 29 |
| 6.6.4. | Column Decimal Places | 29 |
| 6.7. | Authority | 29 |
| 6.8. | Default | 30 |
| 6.9. | Null Value | 30 |
| 6.10. | Derivation | 30 |
| 6.11. | Description | 30 |
| 6.12. | Notes | 30 |
| 6.13. | Allowable Values | 30 |
| 7. | ATTRIBUTE DEFINITION STANDARDS | 31 |

| | | |
|------------|--|-----------|
| 7.1. | General | 31 |
| 7.2. | Attribute Name | 31 |
| 7.3. | Redundant attributes of subtypes | 31 |
| 7.4. | Attribute Comment | 32 |
| 7.5. | Attribute Domain | 32 |
| 7.6. | Average Length | 33 |
| 7.7. | Attribute Optionality | 33 |
| 7.8. | Attribute Volume | 33 |
| 7.9. | Null Value | 33 |
| 7.10. | Default Value | 33 |
| 7.11. | Allowable Values | 33 |
| 7.12. | Attribute Description | 33 |
| 7.13. | Attribute Notes | 33 |
| 8. | DIMENSIONAL MODELING CONSIDERATIONS | 35 |
| 8.1. | General | 35 |
| 8.1.1. | Definitions | 35 |
| 8.1.2. | Facts and Dimensions | 35 |
| 8.1.3. | Dimensional elements related to Entity-Relationship models | 35 |
| 8.1.4. | General Standards for Modeling Dimensions and Facts | 36 |
| 8.2. | Modeling Dimensions | 36 |
| 8.2.1. | Hierarchies | 36 |
| 8.2.2. | Dimension definitions | 37 |
| 8.3. | Facts | 38 |
| 8.3.1. | “Measureless” facts | 38 |
| 9. | MODELING FUNCTIONS, RULES, AND DYNAMIC BEHAVIOUR | 39 |
| 9.1. | Aspects of functional and dynamic modeling | 39 |
| 9.2. | Function, process, and function hierarchy structures | 39 |
| 9.3. | Architectural levels of Function model | 40 |
| 10. | BUSINESS FUNCTION DEFINITION STANDARDS | 42 |
| 10.1. | Terminology | 42 |
| 10.2. | General Standards for Modeling Business Functions | 42 |
| 10.3. | Label | 43 |
| 10.4. | Short Definition | 44 |
| 10.5. | Master Function | 45 |
| 10.6. | Elementary | 45 |
| 10.7. | Frequency | 45 |
| 10.8. | Response Needed | 46 |

| | | |
|------------|--|-----------|
| 10.9. | Description | 46 |
| 10.10. | Notes | 47 |
| 10.11. | Function Type | 48 |
| 10.12. | Function Calling/Called by | 48 |
| 11. | BUSINESS PROCESS DEFINITION STANDARDS | 49 |
| 11.1. | Terminology | 49 |
| 11.2. | Processes within the function hierarchy | 50 |
| 11.3. | When process models are required as deliverables | 50 |
| 11.4. | A suggested approach for modeling process | 51 |
| 11.5. | The scope of process representation | 51 |
| 11.6. | Functions within the context of processes | 52 |
| 11.7. | Process beginnings and ends | 52 |
| 11.8. | Labels | 52 |
| 11.9. | Short Definition | 53 |
| 11.10. | Elementary | 53 |
| 11.11. | Frequency | 53 |
| 11.12. | Response Needed | 53 |
| 11.13. | Description | 53 |
| 11.14. | Notes | 53 |
| 11.15. | Function Type | 53 |
| 12. | EVENT DEFINITION STANDARDS | 54 |
| 12.1. | General | 54 |
| 12.2. | Event Name | 54 |
| 12.3. | Type | 54 |
| 12.4. | On Condition | 55 |
| 12.5. | Time Events | 55 |
| 12.5.1. | Date | 55 |
| 12.5.2. | Time | 55 |
| 12.6. | Frequency | 56 |
| 12.7. | Frequency Unit | 56 |
| 12.8. | Change Events Entity | 56 |
| 12.8.1. | Attribute | 56 |
| 12.9. | System Description | 56 |
| 12.10. | Description | 56 |
| 12.11. | Notes | 56 |
| 13. | RULE DEFINITION STANDARDS | 57 |

| | |
|--|-----------|
| 13.1. General Approach to Modeling Business Rules | 57 |
| 13.2. Rule classification | 57 |
| 13.3. General Standards and Guidelines for Recording Rules as Functions | 61 |
| 13.3.1. Rules Hierarchy | 61 |
| 13.3.2. Rule descriptions and specifications | 62 |
| 13.3.3. Mapping of Rule Functions to the Data Model (CRUD Matrix) | 62 |
| 13.4. Data Integrity and Quality rules | 62 |
| 13.4.1. Entity level rules | 63 |
| 13.4.2. Relationship Rules | 64 |
| 13.4.3. Domain Rules | 65 |
| 13.4.4. Attribute Rules | 65 |
| 13.5. Value Transition Rules | 66 |
| 13.6. Operational Integrity Rules | 67 |
| 13.6.1. Operation Constraints | 67 |
| 13.6.2. Action Triggers | 70 |
| 13.6.3. Process Constraints | 70 |
| 13.7. Business Control Constraints | 71 |
| 13.7.1. Audit Logging Rules | 71 |
| 13.7.2. Authorization Rules | 72 |
| 13.7.3. Balance Control Rules | 73 |
| 13.8. Information Integrity Rules | 73 |
| 13.8.1. Derivation Rules | 74 |
| 13.8.2. Data Matching Rules | 74 |
| 13.8.3. Data Transformation Rules | 74 |
| 13.9. Function Detail Rules | 74 |
| 14. APPENDIX A – ORACLE RESERVED WORDS | 76 |
| 15. APPENDIX B – ATTRIBUTE CLASSES AND DOMAIN NAMES | 83 |
| 15.1. Standard class words and Root Domain names | 83 |
| 15.2. Class word assignment guidelines | 86 |
| 15.2.1. Identifiers and other types of labels | 86 |
| 15.2.2. Binary (yes/no) conditions versus Codes / Characteristics | 86 |
| 15.2.3. Ordering/ranking schemes | 86 |
| 15.2.4. Numeric values | 86 |
| 15.2.5. Computed/derived/summary values | 86 |
| 15.2.6. Alphanumeric strings | 86 |
| 16. APPENDIX C – EXAMPLE RELATIONSHIP NAMES | 87 |

1. Introduction

This document is maintained and distributed by the Data Administration section of the Information Management Group (IMG). The IMG is part of the Information and Technology Management Branch (ITMB), which provides services to the Ministry of Education, Ministry of Advanced Education, Training and Technology, Ministry of Labour, and the Industry Training and Apprenticeship Commission (ITAC).

1.1. Purpose of this document

This document outlines the Ministries' standards for developing information system requirements in Designer. This document is not a tutorial on how to use Designer, and assumes that the reader has a working knowledge of the tool. The document is also not a full description of the Ministries' development methodology, although it reflects many of the strategies currently being pursued for Project Management, Project Planning and Deliverable Management, and Data and Information Management.

1.2. Audience

The audience for this document is all analysts (technical and business), business experts, and consultants performing requirements definition work at the Ministries.

1.3. References

1.3.1. IMG Documents

The following documents, many of them working papers at the time of publication, were prepared by IMG and are available on the LAN or from the IMG staff. They are complementary to this document:

- Data Administration Charter
- Data Architecture Framework
- Repository Management
- Configuration Management
- CDM deliverables for projects

For access to the enterprise and / or reference data models in the Designer Repository, please contact IMG staff responsible for Data Administration or Repository Management.

1.3.2. Books and web resources

The following resources provide additional information about some of the topics mentioned here. Most of the books are available through the BC Government IT Library.

Entity-Relationship modeling:

- CASE*Method: Entity Relationship Modeling (Barker)
- The Data Modeling Handbook (Reingruber & Gregory)
- Data Modeling Essentials, 2nd edition (Simsion)

Generic Data Models:

- Data Model Patterns – Conventions of Thought (Hay)

- Data Model Resource Book (Silverston, Inmon, Graziano)
- Analysis Patterns (Fowler) (N.B. this uses object-oriented notation, which represents logical data structures in a different way from the E-R notation used here)

Modeling Business Rules:

- Business Rule Book: Classifying, Defining and Modeling Business Rules, 2nd edition (Ross)

Dimensional Modeling:

- The Data Warehouse Lifecycle Toolkit (Kimball), particularly section 2, Data Design (N.B. despite the way Kimball positions E-R modelling, we are using his approach to Dimensional modelling in conjunction with E-R modelling; see chapter 8 **Dimensional Modeling Considerations**).

Enterprise Architectures:

- Building Enterprise Information Architectures (Cook)
- Zachman Framework (<http://www.frameworksoft.com/html/zifa.html>)

1.4. Change History

| Date | Version | Author | Changes |
|-----------------|-------------|------------|--|
| June, 1999 | 1.0 | Data Admin | Adapted from the “Requirements Modeling & Specification, Guidelines and Standards” document in use at the Ministry of Health as at June, 1999. That document was originally authored by S. McMillan in July, 1997 and revised by G. Power in March, 1998. |
| September, 2001 | 2.2 | Data Admin | Substantial revisions, to reflect the following: <ul style="list-style-type: none"> • Increased emphasis on abstraction and generalization in data modeling • Use of general reference data models and enterprise models • Preliminary inclusion of decision support requirements modeling guidelines, including dimensional modeling • Updates to Oracle’s Custom Development Method (CDM) version 2.6.0 • Updates to the Ministry of Health document originally used as a base (now version 2.3 as of October 31, 2000) |
| July 2003 | 3.0 Draft 1 | Data Admin | Substantial revisions, as follows: <ul style="list-style-type: none"> • Reorganization of overview sections, and addition of an overview chapter on modeling |

| | | | |
|----------------|-----|------------|--|
| | | | <p>functions and processes, to provide more coherent background and relate modeling to the scope and evolution of information systems architecture.</p> <ul style="list-style-type: none"> • Various updates to sections on data modeling, to reflect guidelines and experiences related to the use of GAME. • Miscellaneous other changes and updates throughout. |
| December, 2003 | 3.1 | Data Admin | Major revision of the section on Rule Modelling. |
| April, 2004 | 3.2 | Data Admin | <ul style="list-style-type: none"> • Various updates to include consideration of object-oriented approaches, including support for some aspects of Use Case definition in function models • Clarification of the sections on Event modeling, and Appendix B on Domain and attribute naming. |

2. Background

2.1. What is modeling?

Methods of abstraction are used in many disciplines to represent real-world situations. Models are used extensively in the context of Information Management and Information Technology (IM/IT), as structured representations of requirements, designs, and implemented system components.

Requirements modeling refers to the formal definition of business activities, and the information required to support those activities, for the purpose of developing systems. Techniques for requirements modeling include process modeling, dataflow diagramming, entity relationship modeling, state transition modeling, function hierarchy modeling, business rule modeling and others.

Models are useful for the representation and organization of business requirements, analytic discoveries and decisions, and system design. Their usefulness stems from a number of characteristics, particularly the following:

- They involve principles of consistent content and structure, which are useful in assuring clarity, thoroughness, and robustness in the work products. Patterns of consistent representation, and consistent thinking about the process of representation, help ensure that important aspects of the different elements of the model are thought through and understood consistently. Manual and automated processes can also be used to analyze the formal representations and check for omissions and anomalies. However, neither the modeling techniques and tools themselves, nor the adherence to standards and guidelines, are sufficient to guarantee these qualities. This requires the engagement and commitment of all parties involved in defining and reviewing models – but the techniques, standards and guidelines help.
- They lend themselves to representation in diagrams, which provide a synoptic view of the model contents, for communication and discussion among the people involved in the various aspects and stages of system definition. (Often people think of the diagrams themselves as the “models”, but the content and long term value of models is largely related to supporting text and characteristics of the elements of the models. Very little of this document, for instance, is concerned with diagramming.)
- As abstractions, they promote the evolution of generalized approaches and solutions, and help to establish solid common patterns of system structure and design that are adaptable to business variation and change.
- As formalized information structures, which can be stored in a data base or repository, they can be used by a variety of software tools, to perform functions such as change impact analysis, model validation, or code generation. The collection of stored models is an essential part of the system and information management resource we call *metadata*. To supplement Repository Management, Configuration Management techniques and tools can also make it practical to maintain traceability for changes and variations in models.

In order to meet their strategic information management objectives, the Ministries require that business requirements be expressed through fully integrated data and function models. These models replace, or rather formalize, the detailed textual descriptions in ‘business requirements documents’ of former years. In general, these models consist of –

- Data models, comprising definitions of entities, relationships, domains, and attributes; and
- Function models, including function and rule definitions.

They may also include process models and event specifications.

All the components of these various models are represented as specific types of objects in the Oracle Designer Repository. The Repository also includes objects that define aspects of the business context, such as Objectives and Business Units. These are not explicitly covered in this document, as they are not really models as such; however, they may be referred to from time to time as they can have important associations to the components of specific models.

The Repository is a data base for the concise and effective storage of these models, and other aspects of requirements and design specifications. It allows for automated tools to validate model integrity and other aspects of quality, management of the traceability of requirements and designs, printing of reports for review or analysis of the repository contents, and impact analysis when business or system changes are considered. It also allows for the sharing of model assets, and configuration and change management of requirements specifications. These are all generic benefits of the use of a repository; specific aspects of the Designer repository are discussed in more detail in section 2.6.2, further below.

The models are owned by the Ministries and administered by IMG.

2.2. Scope of modeling discussed in this document

As mentioned in the previous section, we are dealing here primarily with models of data and function, and also the relationships between these, and their relationship to people and organizations. We are also dealing with business and system rules, with a particular emphasis on rules related to the management of data.

Although the focus is primarily on business requirements modeling, aspects of this document are also concerned with models for purposes of system planning, and the adaptations of business models that result from subsequent systems analysis. The full scope of the suite of models is depicted in the chart below.

| Architecture Level | Data | Rules | Functions | Interfaces |
|--|--|---|--|---|
| Enterprise Planning Objectives, Scope, Context | Entities and Relationships; Domains and Attributes as Standards | [high level business drivers and key events] | High level function model | [Model of stakeholder / user community] |
| Requirem'ts Model of the Business | Entities, Relationships, Domains, Attributes | Preliminary Rule Hierarchy | Business Function Hierarchy | Map functions to User Groups |
| Analysis Model of the Information System | Tailoring of model towards physical DB | Detailed Rule (Function) Hierarchy articulated as Events, Conditions, Actions | Non-interactive functions (e.g. batch; system controls) | Interactive system Functions, articulated as use cases |
| Design Model of the internal structure of the Application System | Tables, Columns, Schemas, Indices (DB Standards) | Rule implementation and Packaging into modules; see also functions. Many rules need to be specified in PL/SQL. (GAME; PL/SQL Stds) | Operations manifest through API's; Modules bind rules. (GAME; Java; Application Architecture (AA) TBD) | UI Tool specific (Oracle Forms; JSP; HTML and XML; others TBD in relation to AA) |

This chart is largely based on an adaptation of John Zachman's framework for Information System Architecture, with which many readers should be familiar. This document is primarily concerned with the Requirements and Analysis rows, with some discussion of the Enterprise Planning row. The formats of models at these three rows are similar, and progress down the three rows is expected to be evolutionary. (The Design Row is included primarily to provide context; see below.)

- The Enterprise Planning level is largely conceptual, and intended to provide a context for planning. Effective models at this level need to be implemented through strategic, high level business analysis projects such as Information Systems Plans (ISP's), and revisited on a regular basis, at least yearly. They should be maintained as on-going definitions of the business that can be used as a context for planning business and system change, and particularly for strategic management of the information resources and application portfolios. The high level models of business functions and of the stakeholder community should be maintained, if possible, in accordance with the principles laid out in this document, but the most urgent are the enterprise level data models of each Ministry, which are maintained at a relatively detailed level. These are discussed in more detail in section 3.2 Use of Enterprise Data Models.
- The models at the Requirements level are structured representations of the business data, functions and rules that need to be supported by one or more systems. It includes clear business definitions and explanations, based on input from business representatives, and approved by them. The Requirements should also be formally aligned with the Enterprise Planning models, with any misalignment clearly documented and explained. It may be necessary to adjust the Enterprise Planning Model to reflect new perspectives that have arisen during the Requirements modeling for a new system, particularly a system with a large scope or strategic business relevance.
- The Analysis models are extensions of the Requirements models, showing additional detail that can be combined with the Requirements models to present a preliminary view of the system design in the business context. Function specifications are refined down to the level where individual system interfaces can be defined and related to the actual business tasks, and data specifications are extended to include structures required to hold intermediate or derived data such as interface queues or report data. The analysis model structures – commonly referred to as the System Data Model and System Function Model – are expected to be maintained to reflect system changes, so that they can continue to be used for traceability, and for understanding of the system in its context. It is important to recognize that the requirements models actually evolve into the analysis models, and do not have a continued existence as independent structures. Therefore, it is essential that any aspects of the business definition, as well as traceability to the Enterprise Planning models, are preserved in the analysis models. (Where GAME is used in an application system, aspects of the analysis model are defined and stored in GAME metadata. See section 2.6.4, Generalized Application Management Environment (GAME).)
- The Design models represent the elements and internal structure of the system as it will be – and eventually is – implemented. This is beyond the scope of this document, and also, to an increasing extent, beyond the scope of Designer as we are now using it. This level is shown here for context, since Design models typically involve different formalisms that relate directly to the implementation architecture, and may be defined and maintained through a completely different tool set. Annotations on the Design level identify tool sets and related architectural areas where standards and guidelines already exist or are in preparation.).

Note that these are described here as levels of model, not as project phases. This document is not intended as an outline of a System Development Life Cycle (SDLC). While the levels of model map fairly closely to phases in a “waterfall” SDLC, such as the one defined for Oracle CDM Classic, the levels of model may also be pertinent to time boxed and iterative models. In particular, the IMG standard is that all projects implementing business application systems **must** deliver data and function models in Designer at the analysis level, and relate these back to an Enterprise Data Model where one

exists, regardless of the development methodology or approach. The standards and guidelines in this document are pertinent to all application development projects undertaken for the Ministries.

There may be some ambiguity around what we consider an *enterprise*. In the context of this discussion, we can think of an enterprise as an organizational structure with a clearly defined purpose or purposes, and clear lines of management accountability and control. In general, each of the Ministries supported by ITMB is considered an *enterprise*, but we may have business reasons for considering enterprise planning over a broader – or, in some cases, a narrower – scope, depending on the focus of senior client management.

In some cases, it may be useful to think of the planning, modeling and development as actually being done for a *federation* of enterprises with a number of common or shared objectives, assets, and processes, rather than as a single enterprise. The group of enterprises supported by ITMB can be thought of as a federation. There are other important federations to consider as well: for example, the combination of School Districts and Schools with some or all program areas in the Ministry of Education may be considered as a federation, as may the combination of Post Secondary Institutions and the program areas of AVED that are concerned with post secondary education. From our perspective, a federation has requirements for sharing and consolidating data and other information resources, similar to those of an enterprise; but the governance structure of a federation is collaborative, as opposed to the more hierarchical governance structure common among enterprises. It may be appropriate to treat a federation as an enterprise from the perspective of modeling and strategic planning for information management, information systems, and / or information technology. For data architecture and information management in particular, because of the critical importance of effective information and information exchange across the individual sectors, we tend to take a broader view of scope than might be assumed by looking at the functional scope of the Ministries in question.

2.3. Information Management Context

Information Management is a set of related disciplines that involve providing for and monitoring various aspects of the data stored in data bases and files, and its usefulness in providing information for legitimate business purposes. Among the aspects of data and information that come under this heading are definition, clarity, integrity, quality, availability, and security. Information Management is of major importance to the Ministries, because of the extent to which many of the core functions of government – policy and legislation, monitoring and governance, and funding – are dependent on good and current information.

The approach to modeling put forward by this document is closely aligned with the priority of information management, in two important ways:

1. It emphasizes an early focus on data, putting a particular emphasis on the priority and detail of data modeling at all three of the Enterprise Planning, Requirements, and Analysis levels. For instance, the enterprise planning level of data model (or Enterprise Data Model) should be defined in sufficient detail that a subset can provide the first draft of the Requirements data model for many types of application systems; and the Analysis Data Model (or System Data Model) should effectively present a logical design of the data base.
2. Through use of the Repository, it emphasizes effective management of the content of the models themselves, which is metadata.

The principles, guidelines and standards presented in this document are presented in the context of the following strategic information management objectives:

- To improve service to customers by providing reliable information when and where it is needed;

-
- To provide efficient and flexible support for service operations and business processes in a rapidly changing business and technical environment;
 - To enable well-informed business decisions at all levels;
 - To reduce the impact of business change on systems maintenance cost.

To achieve these information management objectives, the Data Administration unit is working with ITMB and the Ministries to pursue the following strategies:

- Establish registries of strategic information required by operational units throughout the Ministries;
- Establish Decision Support Environments where information is consolidated, data quality is assessed and improved where possible, and information products and tools are provided to support business information needs in a consistent but flexible way.
- Develop an information architecture which is resistant to business and technology change;
- Establish and administer a fully integrated repository for documentation of business requirements models and system design specifications;
- Define, maintain, and enforce standards and guidelines for business requirements and analysis modeling, and system design specification.

2.4. Fundamental Principles of Requirements Modeling

The Ministries' approach to requirements modeling reflects the Ministries' requirement for a fully integrated, consistent approach to developing business support systems. Requirements definition projects must embrace the following principles:

1. Business requirements definition and modeling is the responsibility of the Ministries. While it is recognized that requirements definition may be undertaken in partnership with external agencies, the ultimate responsibility for the quality and completeness of the requirements models lies with the Ministries.
2. Business requirements are to be expressed in the form of *fully integrated* data and function models. These models are to reside in the central repository.
3. Textual documentation is to be integrated with the models as appropriate, and as described in these Standards and Guidelines. Additional documentation, not necessarily part of a formal requirements model, should also be provided to cover such system requirements areas as non-functional requirements (e.g. for availability and recoverability), critical business constraints on project implementation, etc.
4. Requirements models must be based on models at the enterprise planning level, where these exist. Where there are no enterprise level models, a reference model can be provided as a starting point by IMG.
5. Business policies and processes are subject to change. All data and function/process models are to be developed at a level of abstraction and generalization that allows for changes in business policy and practice without compromising the fundamental models.
6. The validity of information changes over time. The data model must allow for changes in the validity or currency of time-variant information.
7. All information required to provide service to customers is to be expressed in the data model. This includes records of service requests and evidence of the status of a request as it flows through the processing cycle.

(It is important to note that the *status of the request* is not to be confused with the *status of the process*. The status of the request (e.g., 'received', 'approved', 'rejected', etc.) is modeled as data; the events and associated processing activities that cause this status to change are to be expressed in the event and function/process models. The relationship between a processing activity and the resultant state of the data can be expressed through cross-references between elements of data and specific functions or processes.)

8. Where information is required for process measurement and executive decision-making, this information is to be expressed in the data model, regardless of how support for the business process workflow is implemented.
9. Due to the higher level of generalization on the data model (or supertyping, see section 3.5.1), the ability to rely on referential integrity for enforcement of business rules will be reduced significantly. This document includes standards for the formal expression of rules not enforced in database structure. It is expected that a data-driven rule enforcement mechanism will be used to support changing business rules with minimum impact on system structures.

2.5. Approach to Standards in the context of Oracle Designer

The standards proposed here are based on the use of Oracle Designer 6i, for requirements modeling. They are designed to take advantage of the Designer metamodel, and are intended to allow for choice in implementation architecture. Elements that can be defined as structured properties of the data model in Designer are expressed as elements of the data model. This allows these elements to reside as discrete, editable objects in the repository. Similar advantage is taken of the function modeling capabilities of Designer, and the same approach may be used for event and process modeling.

These standards depart from regular use of Designer in several important respects:

- The definition of those types of business rules that Designer does not support directly in a structured manner. Such rules are to be expressed as functions in a separate Rules hierarchy, as outlined in section 13 Rule Definition Standards, or in the GAME repository, if GAME is used during analysis phase.
- Some structural properties of data, which are not reflected in the Designer metamodel, are covered by special notations in the Notes section of the entity definitions. This is evident in the chapter on Entity Definition Standards, for Time Dependency and Unique Identifiers. These approaches do not need to be used if equivalent information is being captured in GAME.
- A number of aspects of model definition, particularly the data models and rule definition, are being adjusted to integrate with the use of GAME (Generalized Application Management Environment, see below).
- Conventions for definition of the system function model are being tailored towards alignment with the definition of use cases in UML. The use case approach has proven itself as very effective for establishing specific user interface requirements, whether or not other object oriented techniques are used, so we are starting to advocate wider use of this approach, to the extent that we can support it in Designer.

We are starting to look at alternatives to Designer, and are no longer expecting design details to be captured in Designer except for the data base designs, and for user interfaces where Oracle Forms or hand-coded PL/SQL is used. However, we are still intent on working with Designer as a repository of business and system function definitions, and of data models from the enterprise level through to the definition of tables. It is clear that there are issues with Designer, but we believe that the integrity of the model information across the enterprise – and of the data model information in particular – is of sufficient value to outweigh these tool-related concerns, at least for the moment.

2.6. Operational Infrastructure

2.6.1. Requirements Modeling Tools

2.6.1.1. Oracle Designer

Oracle Designer (version 6i) is the tool presently selected for requirements modeling for the Ministry. Standards presented in this document are based on the use of the Designer repository for requirements documentation.

(As of July, 2003, because of technical problems with the performance of Designer 6i for very large data models, some of the models are still in Designer 6.0. In most respects, the standards and guidelines in this document apply equally well to both releases.)

2.6.1.2. Other Diagramming Tools

Other diagramming tools, such as Visio or PowerPoint, may be used for diagrams not supported by Designer (e.g., state transition diagrams), and for Process Context diagrams (see the chapter on Process later in this document). Visio is the preferred alternate tool for this purpose. Diagrams prepared using tools other than Designer must be referenced as 'Documents' within the Designer repository.

2.6.1.3. Text Documents

Documents produced to supplement the Designer documentation are to be prepared in MS Word, and referenced as 'Documents' within the Designer repository. Any other electronic files used to supplement Designer must be similarly stored and controlled within the Repository for purposes of version and configuration management.

2.6.2. Central Repository

Oracle Designer stores declarative and narrative descriptions of all elements in the models discussed in this document, in an Oracle database, called the *repository*. In Designer 6i, this repository is integrated with Oracles Software Configuration Manager (SCM). The contents of this database can be retrieved in the form of either reports or diagrams. To ensure consistency and integrity of information across the enterprise, the information systems development process is supported by a central shared repository, owned and administered by the Information Management Group (IMG) of the Information and Technology Management Branch (ITMB).

The central repository will be the authoritative source for definition of the Ministries' information resources. This repository will house all the following system documentation:

- business requirements expressed as fully integrated data, business function and business process models
- database design specifications, which map directly to information requirements expressed on the data models

For some application systems, in particular for any applications developed using Oracle Forms, or using structured, hand-coded PL/SQL, the repository will also contain the application design specifications, which map directly to functions expressed in the function and process models.

2.6.3. Repository Management

The Information Management Group (IMG) is responsible for the administration of the repository, for the quality of its contents, and for the preparation of policies, guidelines and standards regarding its use.

Access to the central repository will be made available, as appropriate, to systems development and maintenance projects. Ownership of elements within the repository is managed on a controlled basis.

The disciplines for using the Designer 6i repository include not only Repository Management but also Configuration Management. The latter includes standards and processes for version control of individual elements and groups of related elements, and orderly promotion and sharing of these configured sets.

2.6.4. Generalized Application Management Environment (GAME)

GAME is a software environment for capturing analysis and design metadata, particularly the metadata related to models of data, rules, and various aspects of the use of data. The data model portions of this metadata can be populated from Designer and populated back into Designer.

GAME is built on a powerful extended model that includes aspects of rule modeling, active database design, and object-oriented analysis and design, as well as more conventional information engineering. It can be used to generate the code to run in the database tier and manage data integrity and quality, as well as other functions related to data synchronization and the delivery of Decision Support data structures and services. Our standards allow for the analysis level modeling to be done in GAME for data, and for data-related events and rules, provided that the data model elements are synchronized back into Designer (there is an automated tool to do this). GAME does not currently support system function modeling, or the specification and generation of user interface capabilities.

3. Data Modeling Standards

3.1. General Concepts

Data modeling is a disciplined method for describing the structure and meaning of business information. The term, as used in this document, refers primarily to entity-relationship modeling. A *data model* is a formal abstraction of the elements of information used or created in a particular group of business functions or activities.

The terms *entity-relationship model* and *entity-relationship diagram (ERD)* are not synonymous either in context or in scope. The model contains much more information than is displayed on the diagram, including full descriptions of each entity and its attributes, and links to the business functions that use the entity. A single entity defined in the model may be represented graphically on several different entity-relationship diagrams, just as they may be represented on several different reports.

In the context of this document, the term 'data model' refers to the descriptive and structured contents of the repository. Where a diagram is discussed, the term 'diagram' will be used.

In general, data requirements for systems development and maintenance must be modeled using entity-relationship disciplines and diagrams. The one exception to this is the use of dimensional modeling. A dimensional approach may be used to represent business data requirements where the following conditions apply:

- The system under consideration is to be a data mart for support of ad hoc query and possibly some reporting for a clearly defined set of users and business domains.
- The required data has already been represented in a base entity-relationship model, or there is a concurrent and interdependent project under way to do so.

For more detail about dimensional modeling, see the chapter entitled Dimensional Modeling Considerations.

The specifics of standards and guidelines relating to the data models are discussed in more detail in chapter 3 Data Modeling Standards, and in the specific chapters dealing with Entities, Relationships, Domains, Attributes, and Rules.

3.2. Use of Enterprise Data Models

ITMB's strategies for Information Management and Data Architecture emphasize enabling the following capabilities:

- The sharing of operational data created or used by different business areas, as appropriate for business requirements, through shared data bases.
- The effective consolidation of data from different data bases into data warehouses or data marts of sustainable data quality, for purposes of reporting, analysis and/or research.
- The standardization of common, flexible design and implementation for widely used patterns of data structures (such as addresses and accounting structures) and for the low-level program logic that manipulates them. This is intended to improve the effectiveness of application system implementation and maintenance, through the reuse of these adaptable patterns.

An essential aspect of the strategic framework for enabling these capabilities is the use of enterprise data models, at the level of the major enterprises and/or federations supported by ITMB. Conventions and structures for sharing specific data across application systems in the relevant enterprise or federation scope are explicitly defined in these models. The inclusion of data elements (entities,

attributes, relationships, or domains) in an enterprise data model is dependent on whether or not they are viewed as *strategic*:

- **Strategic data elements** that are considered critical to the successful operation of the enterprise as a whole, will be defined and maintained as part of the Enterprise Planning Data Models (Enterprise Data Models or EDM's) of the Ministries. The maintenance of these enterprise data models is the responsibility of the Data Administration Unit in IMG with the direction of the appropriate business clients, although it may be contracted out with the agreement of both IMG and the business. The fundamental structures and descriptions of strategic entities, attributes relationships and domains will not be permitted to be altered except by authority of the Data Administration unit.
- **Non-strategic data elements** that may be shared between operational areas, but are not considered critical to the operation, decision making, or analysis of the enterprise as a whole, will be defined in the requirements and or analysis data models for the relevant application systems. These data elements and models may be owned by operational application systems. They will be stored in the repository, and managed by the responsible project or maintenance teams, with the direction of the Data Administration unit.

Where an application system model changes or replaces a structure defined in the pertinent enterprise data model, this difference must be documented and explained in the notes on the relevant structure in the application system model. However, new entities, subtypes, and relationships can be added freely, if they do not overlap with structures defined at the enterprise level. The enterprise model may also expand or evolve because of new understanding developed in the context of a specific application system model. However, it is preferable to review and adapt the enterprise model prior to the data modeling for an application system development project, or to involve the Data Administration unit in doing this as a preparatory stage of the application system data modeling

At a more general and abstract level than the Enterprise Data Models, Data Administration has also defined, and periodically updated, a Reference Data Model, which contains generic structures that can appear and are adapted in the Enterprise Data Models. In the absence of a relevant Enterprise Data Model, Data Administration may be able to provide a version of this model as the starting point for a project. In addition, for any project using GAME, there is a "seed" data model that includes a small number of basic data structures that are assumed by the GAME infrastructure, and can be extended for any application system.

[Note: as of July, 2003, there is a fairly comprehensive Enterprise Data Model for the Ministry of Education, and a project starting up to develop a similar Enterprise Data Model for the Ministry of Advanced Education. There is no equivalent for the Ministry of Skills Development and Labour, although there is a data model developed for the Employment Standards Branch, which shares some of the same structures and characteristics as the EDM for Education, and has strong possibilities of extension (as well as some more immediate relevance) to the rest of the Ministry. The Reference Data Model is currently out of date.]

3.3. Data Architecture Strategies

The data architecture adopted for the Ministries' strategic information is designed to allow for differences in business policy or processing requirements between operational areas, as well as for changes in business policy or process over time. The salient features of this architecture include:

- A level of abstraction which accommodates differences and changes in policy, and should allow for rapid adaptation of systems solutions once they are built;
- Generalization of data structures to allow for greater consistency in processing and data usage;
- Incorporation of the element of time into the data structures, to allow proper management of time-variant data;

-
- Representation of customer service requests as important and persistent information on the data model, to provide improved customer service and management decision making capability.

3.4. Normalization

Entity-relationship modeling has the benefit of being based on principles of mathematical logic. One of the consequences of this is the ability to specify the degree to which a data model is fully refined. The technique for evaluating and increasing the degree of refinement is known as *normalization*, and degrees of normalization are called *Normal Forms*. The most common forms are 1st, 2nd, up to 5th Normal Form.

Enterprise and Requirements models should be in 3rd Normal Form as a minimum. This level of normalization is commonly described as being one where each attribute of each entity is dependent on “the key, the whole key, and nothing but the key”. Where there are entities with large numbers of attributes, particularly where groups of these attributes are interdependent in their optionality or edit rules (e.g. they must either all be null or all have assigned values), these entities should be further broken down along the patterns of these groups of attributes. This brings the model closer to 5th Normal Form.

Where data structures are less normalized, as they may be in the Analysis model (or System Data Model) as data base design considerations start to be reflected, the Notes on the entity types and/or attributes involved should indicate where denormalization has occurred, and why. This must be done where a portion of a model is in less than 3rd normal form.

The following are the kinds of denormalization that are generally anticipated in the Analysis or System Data Model, and permitted where a sound rationale exists:

- Creation of wholly or partially redundant entity types to contain data for purposes of logging, reporting, queued input or output, or other purposes related to the function and dynamics of the system.
- Creation of redundant or derived attribute types in new or existing entity types for speed of access and processing, or for purposes of fiscal control (e.g. financial balancing).
- Collapsing of two or more entity types into one, where there is a one-to-one relationship between them, or where there is a many-to-one relationship where the removal of the entity type on the “1” end does not represent a business exposure for the specific application business area or for the enterprise as a whole. Most commonly, this will be the case if the removed entity type is attributive (i.e. descriptive, not part of the key hierarchy) and where there is no requirement to maintain it separately at the level of the business or the enterprise.. In some such cases, use of an enumerated domain may provide adequate control at the application system level, provided that the values in the domain are controlled at the enterprise level.

Projects that are limited to defining high level business requirements, or are non-transaction oriented (i.e., Data Warehouse, and Decision Support systems), may have data models that are not as constrained by this normalization standard. The decision support denormalizations, however, are expected to follow different general guidelines for dimensional modeling, which is discussed in a separate chapter of this document.

3.5. Generalization and Abstraction

A strategic information management objective is to standardize the meaning and structure of widely-used data so that it can be used across the enterprise to produce consistent, reliable information to customers, service operations and executive decision-makers. In addition, standardization of structure is intended to enable the reuse of significant pieces of application system design and code. To meet these objectives, a more highly abstracted data model is required. (These general requirements and strategies are discussed in more detail in the Data Architecture Framework of IMG.)

The more abstract the model, the wider its application. In terms of its ability to support a diverse and changing business world, an abstract model is much more robust than a model expressed purely in terms of existing business structures and practices. As the level of abstraction increases, however, there is much less reliance on referential integrity and a greater reliance on evaluative domains and other methods of data integrity control.

There are several techniques for raising the level of abstraction on the data model. The following subsections describe techniques used in the base Enterprise Data Model maintained by the Data Administration unit in IMG, and should be followed and used as integration points by projects building Data Models. Specific guidelines on using these techniques, in relation to the Enterprise Data Model, are included in the sections of this document dealing with Entities, Attributes, Relationships, and Domains.

(For further discussion on generic data models read *Data Model Patterns – Conventions in Thought*, by David Hay, and *The Data Model Resource Book*, by Len Silverston, et al. *Analysis Patterns – Reusable Object Models* by Martin Fowler is also a very useful reference on this subject, but requires some background in the use of object-oriented formalisms, since it does not use Entity-Relationship diagramming techniques.)

3.5.1. Supertypes and Subtypes

Supertype entities are used to generalize and group entities that have some, but not all, attributes or relationships in common (e.g., PARTY groups PERSON and ORGANIZATION). The entities generalized by the supertypes are called *subtypes*. Relationships at the supertype level allow the subtypes to participate as peers in important ways. For example, the legal custodian of a child may be either a person or an organization; invoices may be issued to either a person or an organization, payments may be received from either a person or an organization.

The base Enterprise Data Model defines supertypes for a large number of subject areas, including PARTY, PARTICIPANT, LOCATION, SERVICE, AGREEMENT, and ACCOUNT. It also defines and groups entities that reflect the abstractions reflected in the following subsections.

Where entity types in a business specific requirements or analysis data model are defined as subtypes of entities in the Enterprise Data Model – which should happen for virtually all business specific data models – the supertypes from the enterprise data model should be copied into the business data model, although they do not need to be shown on the business E-R diagram (unless they are considered useful for communication purposes).

3.5.2. Representation of Types

Type definitions can be used to represent general characteristics of entity types, and particularly of subtypes, as data structures. Type definitions can be represented through subtyping on the data model, through TYPE entities, or through attributes.

In business data models, TYPE entities need to be explicitly defined and described only if the types have special attributes or relationships important to the business. If GAME is used, the base model defines a hierarchy of Object Types such that every entity type actually has a corresponding TYPE entity that is a subtype of OBJECT TYPE. These can co-exist, and be redundant with, subtypes; however, the entity name of the TYPE entity must be formed by adding the word 'TYPE' at the end of the name of the base entity supertype.

A TYPE entity has a 1-M relationship to the entity type that it describes.

If the type has no attributes other than its name and description, it can be treated as an attribute in an evaluative domain whose name includes the word TYPE. For example, all receipts and payments of money must be posted to corporate accounts according to accounting rules based on the type of transaction and the type of account. The basic attributes of a financial transaction are date, amount and a relationship to a specific customer. A generalized 'transaction' entity with a 'type' attribute allows for extensions to the list of valid transaction types without changing the data structures, while preserving the essential relationship to the corporate account entity. It also facilitates changes to the rules governing the relationships between transaction type and corporate account type.

3.5.3. Domains

Domains are an essential tool in the generalization, coordination, and use of data models. We use Domains to classify and control data attributes with respect to their value ranges, first of all, but also with respect to their formats and usage. A domain is an essential part of the context of an attribute.

From the perspective of data resource management, we can classify entity types and domains first, and then look at attributes as ways in which specific domains are applied to specific entity types. In doing information analysis or developing infrastructure for decision support, we should be able to make frequent use of domains to help narrow the search for specific data elements, or to help determine if specific mathematical operations make logical sense (for instance, adding two quantities together may make sense if they are in the same unit of measure, but will not make sense if one of them is expressed as a percentage, and the other as a measurement).

Domains can be expressed as subtypes of other domains or domain categories, and the foundation set of domains is relatively small. We use a fundamental categorization of domains that has been adapted from the ISO standard, and is summarized in Appendix B (N.B. this list is also used as a basis for attribute naming standards).

The base Enterprise Data Model is intended to be expanded to define a hierarchy of Characteristics, such that every Domain should actually have a corresponding Characteristic entity that is a subtype in this hierarchy. If it is an enumerable domain, that is, if all its values are specified and used as a list by the business and/or the system, then each of these values will appear as a further subtype of this Characteristic entity, in the same hierarchy. In business data models, CHARACTERISTIC entities need to be explicitly defined and described only if the domains have special attributes or relationships important to the business, other than their conventional use in defining the range of values and format for conventional attributes.

(In GAME, the generic supertype that corresponds to CHARACTERISTIC, and encompasses the enumerable domain lists, is called KNOWN VALUE.)

3.5.4. Generalization of Attributes

Other Generalization of attributes can also be used to provide a more extensible model. For example, there may be some types of financial transaction where a reference to an external artifact is required (e.g., cheque number, invoice number). These attributes could be represented as ‘reference type’ (evaluated) and ‘reference number’ to circumvent the problem of having separate attributes for cheque number, invoice number, etc., and discovering later that some are missing. Of course, if these attributes are not mandatory for all instances of the entity, or if the object may have more than one such value, further normalization may be required.

3.5.5. Rule representation

As the models become more abstract, and fewer rules are supported through direct properties of the model and its elements, we need to look into alternative ways of representing these rules. *Business rules modeling* is rapidly being recognized as an essential aspect of data (and function) modeling. Following and elaborating on the Oracle Custom Development Method (CDM), we have defined a standard approach for documenting business rules as part of the function hierarchy, where they are constraints on the data model but cannot be explicitly represented through data modeling formalisms. A summary of this approach, which is relevant to the documentation of rules in either or both Designer and GAME, is presented in section 13, Rule Definition Standards.

Note that this approach to categorizing rules does not apply well to all kinds of rules. It is not explicitly intended to capture rules around process flow, or complex algorithmic logic or decision logic that may be required within a logistically complex business function, although it could be used for these purposes if necessary. This situation is discussed briefly in the chapter on Rules.

3.6. Diagramming Guidelines

3.6.1. Label

- *Include a label on all diagrams*
- *As a minimum, include container name, diagram name, last updated date.*

3.6.2. Relationship Lines

- *Arrange the entities on an E-R diagram so that the relationships flow from top to bottom, with the “many” end below the “one” end. (This is sometimes referred to as the “live crow” standard, because the crows’ feet are “standing up”.)*
- *Minimize crossing lines and elbows, as these make diagrams less easy to understand.*
- *Keep lines straight relative to the vertical – or, if necessary, the horizontal axis. (This can be done using the auto layout icon, but one or more relationships must be selected to avoid having the whole diagram rearranged. An effective way to select all relationships is to select one relationship, then choose Edit – Select – Select Same Type.)*

3.6.3. Colour

- *Colour should be used to improve the readability of models.*

-
- *The use of colour should be accompanied by a Legend, built as a Text Box, which identifies the meaning of each colour used on a model diagram.*
 - *Colour should not be used as the only means of conveying information, as it is a property of the diagram only, and is not stored in relation to the objects in the repository.*
 - *Colour is particularly useful for showing subject areas on data models, scopes of impact for projects or system increments, or scopes of ownership or responsibility.*
 - *The use of colour is often for a temporary purpose, related to scope discussions or change control. Where this is the case, the colour should be applied to a copy of the diagram, not the base diagram.*

4. Entity Definition Standards

An entity is something the business needs to know about, such as a person, a service request, a contract, a fee schedule, a payment, a business policy. The full definition of an entity includes the following:

- a clear definition of its meaning;
- a complete list of its attributes (i.e. descriptive characteristics; attribute details are covered separately in section 7 **Attribute Definition Standards**);
- its relationships to other entities (relationship details are covered separately in section 5 **Relationship Definition Standards**);
- a complete list of the business functions or processes which either create, read, update or delete occurrences of the entity (otherwise known as the ‘CRUD matrix’; This is prepared using the Matrix Diagrammer);
- rules related to the integrity and management of the entity, which cannot be formally captured in any other way, and are represented, by convention, as functions (rule details are covered separately in section 13 **Rule Definition Standards**).

Our standards also include requirements for explanatory and descriptive notes of various kinds, as described further below.

4.1. **Entity Name**

Mandatory, 40 characters

- *This should be the most appropriate name for the entity from a business perspective, across the full scope of its use and meaning. The following aspects of scope need to be considered in particular:*
 - *For subtypes, the name must make sense within the context of the whole model, not just the context of the supertype. For instance, if an entity type of DESCRIPTION has a subtype specific to expenses, that subtype should be called EXPENSE DESCRIPTION, not just EXPENSE.*
 - *To the degree practical, the name (and entity description) should be clear and unique across the whole scope of the enterprise or federation that may want to use this data, either operationally or for decision support purposes. If there is potential confusion or conflict with the name or definition that might be used in another part of the enterprise or federation, this should be mentioned in the Notes.*
- *The entity name must consist of English words (Canadian spelling), written in full.*
- *Abbreviations and acronyms should be avoided in entity names if at all possible. If abbreviation is required due to space limitations, the abbreviation must be defined as a Business Term. (All abbreviations should be defined as Business Terms)*
- *The entity name must be as short and concise as possible, preferably consisting of no more than three words.*
- *The entity name must be singular.*
- *The entity name must be written in uppercase*
- *The entity name must not contain underscores, punctuation marks or symbols.*
- *Avoid the use of Oracle DBMS reserved words in entity names (See Appendix A). Exceptions may be made for supertypes that will never be materialized as tables.*

4.2. Entity Short Name

Mandatory, 10 characters

An abbreviation for the entity name. (This will be used in physical data base design to generate the table prefix.)

- *The short name of the entity should be as meaningful as possible.*
- *Attempt to limit entity names to fewer than 6 characters.*

4.3. Plural

Mandatory, 40 characters

- *A well-formed plural of the entity name must be defined. This is done automatically by Designer, but plurals should be checked. (The plural of the entity name becomes the name of the associated table when the database definitions are generated.)*
- *In particular, where an entity name is made up of several nouns, the plural must be applied to the noun that is the basis for the identification of the entity. For instance, the plural of 'BILL OF LADING' is 'BILLS OF LADING', not 'BILL OF LADINGS'.*

4.4. Entity Volume Information

Entity volume information should be recorded at the Business Requirements model level, and must be recorded at the Analysis model level. Where it cannot be recorded at the Analysis model level, an issue to this effect should be included in the Notes.

- ***Initial volume** should be recorded for all entities, particularly those with large numbers of occurrences (e.g. more than 100,000), unless it is impossible to provide a reasonable estimate.*
- ***Average** number of occurrences should be estimated and recorded for entities where the volume of occurrences is large.*
- ***Average** number of occurrences should be stated in terms of a fixed time frame (recommended as one year); this time frame should be consistent across all estimate within an application system.*
- ***Annual growth** rate should be entered for large volume entities. Annual growth is recorded as a percentage.*

4.5. Synonym

- *Use entity naming conventions for manually-defined synonyms. (Entity short name and plural are treated as synonyms by Designer, and are not subject to entity naming conventions.)*
- *Synonyms should be used to capture any business terms that are used to refer to the entity in a specific context of organization or data usage.*

4.6. Entity Description

Mandatory, 70 characters per line

- *The description of an entity must fully and unambiguously state the meaning and purpose of the entity.*

-
- *The entity description should generally begin with an indefinite article ('a', 'an') introducing a phrase that will complete a sentence. For example, 'A PERSON is a man, woman or child who is involved in, or of interest to, the business'.*
 - *In the definition that begins the entity description, the words that make up the entity name should not be used to describe the entity, except where they are used to refer by name to other entities in the model.*
 - *The entity description should include well-chosen examples to clarify abstract concepts.*
 - *Important rules which help to clarify the meaning of the entity may be included in the text of the entity description, even though they are also described as functions.*
 - *If an entity name is used in description or notes text, the entity name should be written in upper case.*
 - *To the degree practical, the entity description should be clear and unique across the whole scope of the enterprise or federation that may want to use this data, either operationally or for decision support purposes. If there is potential confusion or conflict with the name or definition that might be used in another part of the enterprise or federation, this should be mentioned in the Notes. This is particularly true for entity descriptions at the enterprise or federation model level, where there may be a difference between the usage in the enterprise context and the usage in some more specific or localized business context.*

4.7. Entity Notes

- *Entity notes should be entered under the following standard headings, in the order shown below:*

TIME DEPENDENCY:

*Record the time dependency characteristics as outlined in section 4.8, **Time dependency**.*

SPECIAL CHARACTERISTICS:

Record any other formal properties of the entity type that may need to be defined, particularly those specifically identified in these standards and guidelines, but not explicitly supported by Designer. Each of these is to be entered in the specific format described in this document, and to be separated from the others by a blank line.

ANALYSIS QUESTIONS:

Record any questions that need to be answered by the business before the entity modeling can be completed. Each outstanding analysis question should start with A? to identify that it is an open analysis question. When the question is answered it becomes an analysis point, identified by A!.

DATA STANDARDIZATION NOTES:

Record any details that may affect the effect use of this entity or its contents in relation to data standardization. These may be such things as –

- ***Known conflicts among client organizations or legacy systems, or known shortcomings or inconsistencies, in the way that data is represented.***
- ***Industry or government based standards that are related to data formatting or codification.***
- ***Organizational and other known initiatives that may exist, and need to be taken into account in deciding on content and related controls.***
- ***Indications that standardization will be required, but there is no known standard or organizational reference point to work from.***

IMPLEMENTATION NOTES:

Record any details that may affect implementation decisions in this area, such as possible requirements to keep redundant or denormalized copies of the data to support performance requirements or special kinds of access. For application system models, it may be appropriate to indicate whether specific entities are out of scope, are planned for specific future releases, etc.

MISCELLANEOUS:

Record general information that will assist in understanding the entity, such as historical information or background notes

CHANGE HISTORY:

Record changes to an entity, stating date, reason for change, change made, and initials of the person who made the change. Record changes in chronological sequence

- *Within each of these sections, notes should be numbered. (Other sections may be defined in future.)*
- *The name of the person who made the note, and the date of the note should be entered in parentheses at the end of each note, e.g. (smcmillan jan13/97).*

4.8. Time dependency

- *Each entity type should be classified according to the level of time dependency of the data it represents.*
- *Time dependency categories are as follows:*
 - **Effective dated:** *Each stored occurrence of the entity type applies to a period of time with an explicit start and end, which do not necessarily correspond to the period of time the record is on the data base. For example, a person lives at a specific address for a specific period of time. We represent the beginning and end of this time period with attributes called EFFECTIVE DATE and END DATE. (Note that in modeling relationships of this kind of entity type, we frequently model them as though we were interested only a single effective date range.)*
 - **Event:** *Each occurrence of the entity type represents something that happens at a specific date, or specific date and time. This date or timestamp is an attribute of the entity.*
 - **Periodic:** *Each occurrence of this entity type relates to a regular time period, such as a fiscal year or month. An example would be an operating budget for a specific fiscal year. One of the attributes of the entity type identifies the time period to which it applies. We would usually expect to find occurrences of data relating to successive time periods in sequence, e.g. operating budgets for the same area for successive years.*
 - **Snapshot:** *Each occurrence of the entity type represents data as it existed at a point in time, usually on a cyclic basis, once per period. The point in time is an attribute of each occurrence of the entity type, or of a specific data collection to which it belongs, such as a table or file. Snapshot data is typically not used in operational data bases, but may be very important for Decision Support. Note that this is different from periodic data, although it may appear similar. The grade in which a primary student is enrolled is usually thought of as periodic data, since the student is in one grade for a school year; but it may be important to recognize it as snapshot data, particularly early in the year, because students can be reassigned to different grades after the year has started. The place where the student lives may change several times during the year, and may be effective dated from the perspective of operational application systems; however, for decision support, it may be adequate to capture this information once at a specific time in the year (e.g. the end of September), so this information would be a snapshot.*

-
- *None:* Occurrences of the entity type have no dependency on time, or this dependency is irrelevant.
 - Time dependency should be indicated through a statement in the Special Characteristics section at the beginning of the Notes for the Entity. It should be the first statement in this section. The statement should read “Time dependency is ...” with the blanks filled in with the time dependency type identified above.
 - For some entity types, such as the home address example discussed above, their time dependency requirements for the operational data base and for decision support may be different. In such a case, the time dependency should be described as follows: “Time dependency is ... for operational use, ... for decision support.”

4.9. Unique identifier

- Every entity must have at least one unique identifier, consisting of one or more attributes and/or relationships.
- In general, every entity type should have an artificial unique identifier (‘surrogate key’) consisting of a single numeric attribute, whether or not there appears to be a natural unique identifier. This type of unique identifier should always be named ‘ID’. If this approach is not used, there should be an explanation in the Notes as to why not.
- Apart from the ‘surrogate key’, every entity type should also have one or more unique identifiers known or visible to the business. We will often be interested in capturing information about “candidate keys”, or combinations of data that may be regarded as unique identifiers from a business perspective, particularly as the surrogate “ID” is a purely technical standard. The business users may also have several possible ways of uniquely identifying a piece of data, and some of these may be subtype specific.
- Elements of business unique ID’s or Candidate keys may be part of the entity definition, either through attributes or through inheritance from the unique identifiers of other entity types to which there is a mandatory many-to-one relationship. Where these are composed of entity attributes and keys inherited through relationships, they should be defined as ‘Secondary UID’s’. Alternatively, they may be defined through Unique Identifier Rules; see section 13.4.1.1 Unique Identifier rules.
- Where a unique ID consists of attributes and/or keys inherited from parent entity types, all attributes and relationships in the unique identifier must be mandatory. Where it is based on a rule, all attributes or relationships used by the rule must either be mandatory or be tested by the rule for their existence.
- A unique identifier defined for a supertype must also apply to all subtypes of that supertype.
- Where there is a hierarchy of supertypes and subtypes, the ‘surrogate key’ ID must be defined on the highest level supertype. Where a business UID is specific to a subtype, it must be defined as a ‘Secondary UID’.
- All attributes and relationships in the unique identifier must be stable. No attribute in the unique identifier should ever be updated, nor any relationship in the unique identifier transferred.

4.10. Enterprise Level considerations

In an enterprise or federation model, the following information is not required:

-
- Entity volume information for less significant entity types; however, this is still required for major business entities types.
 - Implementation-specific notes

See also comments on the clarification of scope and relevance for Entity Name and Description.

4.11. Initial Data Model considerations

In an initial business data model, the following information is not required:

- Entity volume information.
- Time dependency.
- Unique identifiers.

It is expected that the notes will be minimal at this stage.

5. Relationship Definition Standards

A business relationship, or relationship for short, is a named, significant association between two entities. A relationship is always binary, in the sense that it is always an association between exactly two named entities, or between an entity and itself.¹

5.1. Relationship Name

- Relationships are named at both ends.
- Relationship names are written in lower case.
- Relationship names usually consist of an adjective or verb participle, often but not always followed by a preposition. They must NOT begin with an active verb.
- Avoid underscores, punctuation marks and abbreviations in relationship names.
- Use consistent names for relationships with the same meaning (e.g., classifying/classified by; including/part of).
- Appendix C – Example Relationship Names contains an extensive list of examples.

5.2. Optionality

- Avoid relationships that are mandatory on both sides if at all possible. If impossible, document reason.

5.3. Cardinality

- Resolve all M:M relationships. Resolving many-to-many relationships is required to make the model compliant with first normal form, and often leads to additional data requirements, e.g., effective dates.
- Balanced 1:1 relationships are not permitted, except for redundant relationships among subtypes (see 5.6 Relationships among subtypes), where a 1:1 relationship among subtypes is a specialized form of a 1:M relationship involving supertypes.
- Minimum and maximum cardinality are recorded as properties of the relationship if applicable.
- The following special relationships must be commented explicitly in Relationship Notes:
 - $1(o):1(m)$
 - recursive relationships
 - relationships with the same optionality at both ends.

5.4. Transferability

- Transferability of relationships must be defined. Transferability applies only to the 'many' (foreign key) side of a relationship. Designer default is transferable.

5.5. Exclusive Arc

An arc may include only relationship ends which are all of the same optionality.

¹ Barker, CASE*Method Entity Relationship Modeling, page 3-3

5.6. Relationships among subtypes

On a business data model, it is permissible to show a relationship constraint explicitly as a relationship between two subtypes, even if this is actually a specific instance of a more generic relationship between supertypes shown on the enterprise model.

For instance, the Enterprise Data Model may show a generic INTER PARTY ASSOCIATION as a way of linking PARTIES to each other. An INTER PARTY ASSOCIATION is shown generically as having a pair of M:1 relationships to PARTY. Clearly, MARRIAGE would be specific subtype of this generic association. On a particular data model diagram, it may be desirable to show the explicit relationships involved in MARRIAGE.

There may also be a variation in rules or constraints on relationships among subtypes (it may be worth noting that these are also subtypes of relationships!). The participants in a MARRIAGE must be PERSONS, not ORGANIZATIONS, so the subtype of INTER PARTY ASSOCIATION is specific to a subtype of PARTY as well. In addition, there may be only one PERSON participating in each of the two relationships between MARRIAGE and PERSON, which makes these 1:1 relationships, even though the relationships among their supertypes are M:1.

- *Where relationships among subtypes are defined redundantly, the names on the relationship ends of the subtype relationship must match the names on the supertype relationship exactly.*
- *Redundant Relationships on subtypes must be at least as constrained as the corresponding relationships on supertypes. For instance, a relationship may be 1:M on a supertype and 1:1 on a subtype, or optional on a supertype and mandatory on a subtype, but not vice versa.*

On an Analysis Level Data Model (System Data Model), redundant relationships may cause problems if they are used to generate data base designs, as they can result in redundant foreign key definitions. Therefore, the relationships may be removed from the levels of entity type that do not correspond to the desired level of table and key definition. However, in such a case, the following must be observed:

- *If a subtype relationship is removed from the model, and this relationship includes a constraint different from the constraint on the supertype relationship, this constraint must be captured either in a rule in the Designer Rules hierarchy (see 13.4.2.1 **Restricted Relationship Rules**), or in metadata associated with some other supported tool, such as GAME.*
- *If a supertype relationship is removed from the model, the notes on the supertype entity should indicate this.*

6. Domain Definition Standards

A **Domain** defines a set of validation rules, format constraints and other properties that apply to a group of attributes, columns, program data constructs, parameters or data structures.

Allowable values specified for a domain are inherited by associated attributes and columns and other objects. All attributes must be associated with Domains.

Names for Domains, like class names for attributes, are prescribed depending on the type of data being defined. In some cases, the actual domain name is already defined at the enterprise level. In other cases – particularly in the case of codes and identifiers – a root or parent domain is identified, and application-specific domains should be classified as subsets of the specific domains in question. The list of root domain names, and the attribute classes to which they correspond, can be found in Appendix B.

6.1. Name

Mandatory, 40 characters
The name of the domain.

- *A domain name must be singular.*
- *A domain name must be short and descriptive, consisting of no more than three words. Do not use prepositions in domain names, unless the preposition is part of a conventional phrase (e.g. Unit of Measure).*
- *Words used in the domain name must be standard English words, written in full. Abbreviations, other than standard abbreviations, are not permitted unless space limitations demand. Any special abbreviation should be defined as a Business Term in the repository.*
- *Underscores and punctuation marks are not permitted in domain names.*
- *Domains created to represent state in an entity life cycle should include the entity name. Example: The domain representing card status should be named CARD STATUS.*
- *Where GAME is used, each Domain Name should be identical to the name of a specific subtype of KNOWN VALUE in the System Data Model. This will allow GAME to use the specific instances of KNOWN VALUES when validating domain contents.*

6.2. Subset Of

Mandatory for all domains except the root domains identified in Appendix B.

The name of the immediate parent for this domain.

- *Enter this value only where a domain hierarchy is being defined. Domain hierarchies will be defined widely in the reference and enterprise models; see Appendix B – Attribute Classes and Domain Names.*

6.3. Comment

Mandatory, 240 characters
A comment or brief description of this domain.

- *The descriptive comment must apply to all attributes and columns associated with the domain*

6.4. Attributes in Domain

6.4.1. Format

The format for the attributes in the domain.

- Valid formats are:
CHAR
DATE
IMAGE
INTEGER
MONEY
NUMBER
PHOTOGRAPH
SOUND
TEXT
TIME
TIMESTAMP
VIDEO.
- *Do not use VARCHAR2 as a domain format. This and other data types not listed above may be specified as Data types for implementation of the domain; see 6.6.1 Data type .*
- *For numeric values, ignore special formats.*
- *Always enter the number of decimal places for attributes of NUMBER format.*

6.4.2. Average Attribute Length

Optional, 5 digits

The average length for attributes in this domain.

- *Do not record average attribute length for domains.*

6.4.3. Maximum Attribute Length

Mandatory where applicable, 5 digits

The maximum length for attributes in this domain.

6.4.4. Attribute Decimal Places

Mandatory only for domains of format NUMBER or MONEY, 2 characters

The number of decimal places for attributes in this domain.

- *Indicate number of decimal places for all domains of format NUMBER or MONEY*

6.5. Unit Of Measure

Optional, 10 characters

The unit of measure for attributes in this domain

- *Unit of measure should be supplied in all cases where it is applicable.*
- *The unit of measure should be written in full Examples: meters, kilograms, pounds, dollars, hours*

6.6. Columns in Domain

-
- *Domain properties related to columns must be completed before generating table definitions.*

6.6.1. Data type

The format for columns, program data constructs, parameters or data structures in this domain.

- Valid formats are:
BINARY INTEGER
CHAR
DATE
DECIMAL
DOUBLE PRECISION
FLOAT
GRAPHIC
IMAGE
INTEGER
LONG
LONG RAW
LONG VARCHAR
LONG VARGRAPHIC
NUMBER
RAW
REAL
ROWID
SMALLINT
SOUND
TEXT
TIME
TIMESTAMP
VARCHAR
VARCHAR2
VARGRAPHIC
VIDEO.

6.6.2. Average Column Length

Optional, 5 digits

The average length of columns in this domain.

- *Do not record average column length for domains; use the column property.*

6.6.3. Maximum Column Length

Optional, 5 digits

The maximum length of columns in this domain

- *Mandatory for domains with formats CHAR, VARCHAR2, NUMBER, or INTEGER*

6.6.4. Column Decimal Places

Optional, 2 digits

The number of decimal places for columns in this domain.

6.7. Authority

Optional, 10 characters

The level of authority needed to update this domain. The default is ANY.

6.8. Default

Optional, 40 characters

The value that attributes in this domain may be assumed to have if none is provided. It may or may not be the most common value for the attributes or columns in the domain; a wiser choice, if a default is to be chosen, may be the value with the most neutral effect on the business.

- *Do not record default value unless there is absolute certainty that the default will always be the same for all attributes and columns associated with the domain. Default values may be entered at the attribute or column level instead.*

6.9. Null Value

Optional, 40 characters

The value that indicates that the attribute or column in the domain is not present for an occurrence of the entity or table.

- *This property need not be defined until Analysis or Design. Systems implementations may record null values by means of spaces, high values, zeros, etc.*

6.10. Derivation

Optional, 240 characters

The derivation of attributes within the domain.

- *Do not define domain derivation.*

6.11. Description

Optional, 70 characters per line

A brief description of this domain.

- *Enter a description only if the Comment is insufficient to describe the meaning and purpose of the domain*

6.12. Notes

Optional, 70 characters per line

Any notes or comments about this domain.

6.13. Allowable Values

- *Every value in an evaluative list must have a clearly and precisely defined Meaning.*
- *Values in an evaluative list must be unique within the list*
- *Abbreviations of values in an evaluative list must be unique within the list.*

7. Attribute Definition Standards

Attributes are details that further describe an entity, and are the particular information components of interest to the business.

7.1. General

- *Attributes that represent relationships are not entered as attributes in Designer. (Foreign key columns in data base designs are auto-generated from relationships by Designer's Database Design Wizard.)*
- *In particular, if an attribute represents the identifier of an entity that is known to the business but is outside the application cope, the entity in question should be included in the model with a brief definition and a unique identifier, so that the relationship can be used instead of an attribute. In this particular instance, the Unique identifier of the entity in question may be something meaningful to the business, instead of a surrogate key. This approach to modeling will help significantly in the integration of models and data across application systems.*
- *Put an attribute at the subtype level only if you can give good reasons why it does not apply to any of its 'brother' subtypes.*
- *Avoid modeling derived attributes. Where a derived attribute is considered essential for communication purposes, justification must be documented in the Entity Notes. (For the special case of modeling derived attributes on dimensional models, see 8 Dimensional Modeling Considerations.)*

7.2. Attribute Name

Mandatory

Maximum 30 characters.

- *Attribute names must be expressed in clear, unabbreviated English, in as natural a syntax as possible.*
- *Attributes should consist of one or more qualifiers, followed by an attribute class. Attribute classes are defined in Appendix B*
- *Attribute names must be singular and should consist of no more than three words.*
- *Avoid abbreviations in attribute names, other than for standard abbreviations listed in Appendix B. Any other abbreviation used must be recorded as a Business Term and added to the Glossary.*
- *Do not use the same attribute name at super- and subtype levels in the same generalization hierarchy, nor in parallel subtypes, unless the attribute has the same meaning in all cases. For the case where an attribute is defined redundantly at the supertype and subtype levels, see 7.3 Redundant attributes of subtypes.*
- *Do NOT include the entity name in the attribute name, unless it is absolutely required for understanding, or for consistency with business usage (e.g. EMPLOYEE NUMBER is likely more meaningful than NUMBER, even on an entity type called EMPLOYEE).*
- *Avoid Oracle reserved words for attribute names (see Appendix A)*

7.3. Redundant attributes of subtypes

In general, it is not advisable to use the same attribute name at super- and subtype levels in the same generalization hierarchy, or in parallel subtypes. However, because of the extensive use of subtypes in the reference model, and enterprise models, we will allow attributes to be specified redundantly at the subtype level in application systems, for any of the following reasons:

-
- The supertypes are not actually shown on the application system model diagrams, and the attributes need to be clearly visible on the diagrams.
 - An attribute is applicable to more than one subtype, and has the same meaning in all subtype contexts, but is subject to different rules (e.g. optionality, default value) for different subtypes.

Where redundant attribute definitions are used, the following standards and guidelines apply:

- *Redundant Attributes on subtypes must be at least as constrained as the corresponding relationships on supertypes. For instance, an attribute may be optional on a supertype and mandatory on a subtype, but not vice versa.*
- *On an Analysis Level Data Model (System Data Model), redundant attributes will cause problems if they are used to generate data base designs. Therefore, the attributes may be removed from the levels of entity type that do not correspond to the desired level of table and key definition. However, in such a case, the following must be observed:*
 - *If a subtype attribute is removed from the model, and this attribute on the subtype includes a constraint different from the constraint on the supertype attribute, this constraint must be captured either in a rule in the Designer Rules hierarchy (see 13.4.4 Attribute Rules), or in metadata associated with some other supported tool, such as GAME.*
 - *If a supertype attribute is removed from the model, the notes on the supertype entity should indicate this.*

7.4. Attribute Comment

Mandatory

- *Every attribute must be clearly and concisely defined in the 'Comment' field.*
- *If a fuller description is needed, this may be provided as Description text.*
- *If a user interface is being generated from Designer, the attribute comments will become the screen hint text for the fields that correspond to the attributes*

7.5. Attribute Domain

Mandatory

- *Every attribute must be associated with a domain.*
- *All attributes associated with a given domain must have exactly the same format, length and precision as the domain*
- *Prefer a separate entity over an evaluative list domain if more than 50 allowable values exist for the domain*
- *Define the following attribute properties in the domain definition, and use the 'Enforce Domain' utility to populate the properties to the associated attributes:*
 - Format*
 - Maximum Length*
 - Decimal Places*
 - Allowable Values*

7.6. Average Length

For variable length strings only.

- *Define average length as a property of the attribute if the volume of occurrences is expected to be large.*

7.7. Attribute Optionality

Mandatory

- *The optionality of each attribute must be defined (Note: The Designer default is “optional”).*

7.8. Attribute Volume

Optional

- *Attribute volumes should be estimated for all optional attributes in high volume entities.*

7.9. Null Value

- *Do not supply a ‘null value’ for attributes. This should be defined as a domain property, at analysis or design time.*

7.10. Default Value

Optional

The value that the attribute may be assumed to have if none is provided. It may or may not be the most common value for the attribute; a wiser choice, if a default is to be chosen, may be the value with the most neutral effect on the business.

- *Enter default values with restraint.*

7.11. Allowable Values

- *Supply allowable values at the domain level, not at the attribute level.*

7.12. Attribute Description

Mandatory

- *The text of the Attribute Comment should be copied into the Attribute Description, and expanded if necessary to give an adequate description of the attribute.*

7.13. Attribute Notes

Additional textual information that provides context for the attribute.

- *Within each section listed below (Other sections may be defined in future.), notes should be numbered.*
- *The first initial and last name of the person who made the note, and the date of the note should be entered in parentheses at the end of each note e.g., (gpower jan13/2000)*
- *Entered with the following standard headings, in order, with the headings included:*
 - **IMPLEMENTATION NOTES:**
Record any assumptions made that may affect the implementation of this attribute, e.g., derivation conditions

-
- *MISCELLANEOUS:*
Record general information that will assist in understanding the attribute, such as historical information or background notes

Analysis Questions and Change History should be maintained at the level of the entity definition, not the attribute.

8. Dimensional Modeling Considerations

8.1. General

8.1.1. Definitions

Dimensional modeling is an approach to modeling data for the express purpose of designing and implementing data bases for specific uses in Decision Support. These data base designs are specifically intended to support Online Analytic Processing (OLAP) and standard reporting services, and may also be useful for some kinds of data mining. Such data bases are generally referred to as *data marts*.

By definition, a dimensional model has a strong physical design orientation. The structures within it are conceptual definitions of physical data base structures, or *tables* in a relational data base. Because a dimensional model is also used for planning a Decision Support Environment (DSE), at the enterprise or application system level, we are providing guidelines for capturing the dimensional modeling elements in relation to Designer. These are primarily standards and guidelines for mapping a dimensional model to an Entity Relationship model, to control redundancy in data definitions and standards, and to provide a framework for doing operational or legacy data mapping to the target DSE.

Because dimensional structures are essentially physical, there are a number of considerations and characteristics of dimensional models that are not really pertinent for requirements modeling, but may be introduced here for purposes of clarification of the dimensional modeling strategies in general. In the rest of this chapter, comments that relate only to physical design are usually enclosed in parentheses.

8.1.2. Facts and Dimensions

A Dimensional model consists of two kinds of tables:

- a basic table, called a *fact table*, which usually contains arithmetic values that may be used in calculation; and
- a number of tables, called *dimension tables*, that are referenced by foreign key pointers from the fact table, and contain information in specific categories that are useful for analyzing the set of facts.

For instance, a fact table might include detailed expense accounting information, and there might be dimensions relating to the time, responsible organization, geographical location, and category of each expense. The facts might then be summarized and/or analyzed according to various combinations of data associated with the dimensions.

Because a dimensional model is often depicted with the fact table in the centre, and the dimension tables radiating from it in all directions, this kind of model is also widely known as a *star schema*. Sometimes the structure of the dimensions requires (or is best served) by a slightly more normalized physical data structure for the dimension-specific data. A dimensional model organized in this way is often referred to as a *snowflake schema*.

8.1.3. Dimensional elements related to Entity-Relationship models

Dimensional models have a significant focus on data base design, as opposed to being purely logical representations of business data — which is, in principle, the nature of entity relationship models. A dimension that contains one or more hierarchies of categories for data aggregation is clearly a combination of data structures that would be represented as separate, inter-related entities in a normalized model. A fact may correspond directly to an entity — like a dimension, a fact should at least have a level of granularity that corresponds and is mapped to a specific entity — but there may be cases where fact tables include derived data or are denormalized to cluster data from several entities.

Dimensional models may be produced in either of two circumstances:

1. The data model is specific to an application system whose primary objective is the delivery of decision support data bases. In fact, such a model is likely to contain only dimensional and fact tables.
2. The data model is defined at the level of an enterprise or federation, either as the enterprise data model or as a derived variant of the enterprise data model used for data warehouse planning and design.

In either case, it is preferred that the dimensional model be constructed in reference to an E-R model, although this may not necessary if the data is coming from an external source and does not require close integration with other data sources (e.g. statistical population data from the BC Vital Statistics Agency or Statistics Canada). It is not recommended that Designer be used for the actual dimensional modeling. A tool more explicitly designed to support dimensional models should be used, or, failing that, a spreadsheet or some such simple tabular representation, supported by diagrams if so desired.

8.1.4. General Standards for Modeling Dimensions and Facts

- *Dimensions and facts should be represented in an Enterprise or data warehouse level data model where they are being defined as conformed dimensions or facts, i.e. dimensions or facts that will have the same structure in all data marts where they are used. In such cases, the E-R model to which they are related should be an Enterprise level model. (In general, conformed dimensions also need to have the same data content, and should be managed centrally).*
- *All contents of the dimension and fact tables, with the exception of derived attributes, must also be represented as normalized entities and attributes in the separate E-R model to which the dimensional model is mapped.*
- *Names of elements in dimensional structures – levels in dimensional hierarchies or values in facts – should be identical to the names of the corresponding attributes in the base entity type, if they have equivalent meanings. If it is necessary to qualify an attribute name to avoid redundancy or confusion, the name of the source entity should be used as a qualifier.*
- *Descriptions of dimensions and facts should be limited to brief definitions indicating the purpose of each structure, with some rationale for its definition. This should be adequate to clarify why it is being defined, and to distinguish it from other dimensions or facts being defined. Descriptions should not be redundant with the hierarchy definition (see below), or with the descriptions of the source entities.*
- *If an element is derived, it should have a name that does not conflict with an attribute name in the base entity type or types in the E-R model. The element description should describe the derivation.*
- *For planning purposes at the enterprise model level, dimension hierarchies and facts should be defined down to the most granular level practical, whether or not there appears to be a business need for data analysis at this level. This will help with understanding, source data mapping, and ensuring conformance across the different uses of common elements.*

8.2. Modeling Dimensions

8.2.1. Hierarchies

Within a dimension, there is usually at least one set of categories that allow information pertinent to the dimension to be aggregated (summarized, averaged, etc.) at various levels. For instance, time is typically susceptible to being summarized by day, month, and year; expense account codes are typically summarized to several levels depending on the organization's accounting structure; geography may be

summarized from address, through postal code, to various levels of administrative or political area, depending on the purpose. This sequence of categories allowing progressive aggregation within a dimension is referred to as a *hierarchy*.

There may be several hierarchies within a dimension. For instance, there may be requirements to summarize – or *aggregate* – time-related data by operating periods and fiscal years in some instances, by calendar months and calendar years in others, by school weeks, semesters, and school years in still others. In order to work together as a dimension, it is important that each of these hierarchies has the same granularity at the lowest level – for instance, in this case of the time dimension, this might be days; in the case of geography, this might be addresses.

As an alternative to forcing hierarchies into the same dimension, it may be appropriate to define somewhat overlapping dimensions, and the inability to establish a common basic granularity may be an indicator that this should be done. For instance, operational budgeting information will not be available at the level of days, but may be viable at the level of periods; this may suggest the creation of a business cycle dimension, which has some of the same attributes as the time dimension.

However, it is better if they can be regarded as the same dimension, as this provides for the most consistency of interpretation and comparison across different parts of the dimensional model – i.e. across different fact types. It should be borne in mind that the logical definition of a shared dimension does not necessarily require that every actual implementation and use of this dimension in a given fact has to resolve to the most granular level. It is often inefficient and unnecessary to carry data in a dimensional data base much below the levels of aggregation that are of business interest.

8.2.2. Dimension definitions

It is preferable to do the formal definition of dimensions before completing the formal definitions of facts. This is a good strategy for establishing standards for shared dimensions, or conformed dimensions, as they are widely known.

Each level in a dimension is usually equivalent to one of the following structures in an E-R model:

- An enumerable domain, preferably defined as a CHARACTERISTIC or KNOWN VALUE in the model.
- A time period.
- A geographic location or area.
- An organization.

Other types of PARTY, particularly PERSONS, will sometimes appear at the most granular level. There may also be business-specific lists of data – such as educational programs or courses, or specific types of available service – that function very much as though they were enumerable domains, but include additional business related data and controls, and so may not actually appear as domains.

The link between each level of the hierarchy and the next, upwards or downwards, should be explicitly visible through the E-R model. This is one of the reasons that it is preferable to show domains as being equivalent to entity types (see the discussion in section 3.5.3).

- *Each level on a dimensional hierarchy should be related to an entity type on an E-R model.*
- *The mappings between adjoining levels in each dimensional hierarchy should be explicitly related to relationships and/or associative entity structures on the E-R model.*
- *If a dimension has more than one hierarchy, the most granular level of all the hierarchies in that dimension must be the same.*

-
- *If a dimension involves more than one hierarchy in parallel, the dimension definition should be supported by a diagram. This is particularly the case if the dimension has a lattice structure, i.e. if the hierarchies intersect somewhere other than the most granular level – in addition to intersecting at that level.*

8.3. Facts

In its simplest form, a fact structure is logically made up of a set of values or measures – all numeric – related to a number of dimensions. A fact can be viewed as a derived entity, or view, with a number of local attributes – all of which are numeric measures – and a number of foreign keys – all of which relate to the dimensions it refers to, at the most granular level to which it refers. The logical unique key of a fact is usually composed of some or all of these dimensional keys.

The fact definition should include definitions of all values, both those copied from the base model and those derived from it. It should also include all relevant dimension references, at the appropriate hierarchy level. This often implies a set of navigations across the base E-R model.

- *The fact definition must be explicitly based on a “root” entity in the base model, with a clear definition of how derived values are derived, and what relationships are used to navigate to other entities of significance in deriving measures or establishing the dimensional context.*

8.3.1. “Measureless” facts

One of the measures related to each instance of a fact is its existence, which can be represented by the number 1. This is significant, for instance, when the number of fact instances in a category is being counted. Sometimes this countable property is the only measure of interest with regard to a specific fact– and facts of this kind are sometimes called *factless* or *measureless* facts.

- *The “count” measure, with a value of 1, must always be specified for a fact definition.*
- *However, if a dimension is logically identical to a measureless fact table, it is adequate to identify only the dimension at the enterprise level. (For instance, STUDENT may be both a dimension for analyzing enrolments or test results, and a basic fact to be counted. At the enterprise model level, it is adequate to have only one STUDENT structure, and this should be identified as a dimension. However, at the data mart level it is anticipated that the FACT structure variants will have to be defined separately and explicitly, since they will likely require different physical implementations, particularly different ways of handling foreign key values.)*

9. Modeling Functions, Rules, and Dynamic Behaviour

9.1. *Aspects of functional and dynamic modeling*

This and the following chapters focus on the functional and behavioural aspects of models. These are centred around the function hierarchy in Designer, although they have a variety of uses. The following specific aspects of modeling will each be covered in one of the following chapters:

- Representation of the business functions themselves in hierarchical structure, at various levels of detail from the summary level supporting an enterprise view, to the view of a group of activities performed by a single system user to perform a single task. For this detailed level of representation in particular, we are also providing guidelines to support the representation of use cases as Designer functions.
- Representation of the business processes that combine sets of individual tasks in formal ways to accomplish specific objectives, such as the satisfaction of a specific service request, or the completion of a specific stage of a business cycle.
- Representation of the events that trigger processing, at various levels of system conceptualization and representation. Of particular importance for requirements and analysis modeling are the major types of business events – such as client requests – that trigger business processes, and the much finer grain events that trigger the firing of rules. These two levels of event description are clearly related to the modeling of processes and of rules, as discussed here. Other kinds or levels of event modeling can also be supported.
- Representation of rules, with particular emphasis on the rules that enforce and/or monitor data integrity, quality, and security.

9.2. *Function, process, and function hierarchy structures*

There is much confusion between the terms *function* and *process*. The Designer repository does not distinguish between a function and a process. Both are stored in the repository as functions.

There are several different modeling methods, which employ function/process definitions. Function hierarchy modeling, process modeling and dataflow modeling all revolve around function/process definitions, each in a different way. Each of these methods has its own notation and syntax for diagramming. Each has its particular focus and its particular advantages and disadvantages.

The Function Hierarchy Diagram displays functions in a semantic hierarchy, without concern for sequence, priority, or organizational responsibility. The Process Diagram displays functions (here called ‘processes’ and ‘process steps’) in the context of the *events* that cause them to be performed (triggers) or result from them (outcomes). This diagram shows the sequence in which functions are performed, the actors or agents (business units) responsible for performing them, and the flows of data and control among them.

Two separate function hierarchies should be defined for each Application System; a third is optional:

- ***Application Hierarchy:*** *The hierarchy of business functions performed within scope of the Application System. This may also include functions that are ‘not automated’, either because they are to be performed manually, or because they are performed using some other systems.)*

-
- **Rules Hierarchy:** A hierarchy containing static data rules and data manipulation rules. Standards and guidelines for business rule definition are provided in section 13 Rule Definition Standards.
 - **Process Hierarchy:** An optional hierarchy containing those functions defined specifically for the process diagram (decision points, root processes, perhaps manual steps), which would not belong in the function hierarchy). Guidelines for developing this hierarchy are defined in section 11 Business Process Definition Standards. This hierarchy and the associated guidelines should be used only if Designer is being used for this purpose.

9.3. Architectural levels of Function model

Function models, and function hierarchies in particular, can be used at the enterprise planning, requirements, and analysis levels of modeling, though different standards and guidelines apply, particularly with reference to levels of detail.

- *In an enterprise function model, functions should be decomposed to a level appropriate for understanding the major considerations in bounding the scopes of application systems within the enterprise. The hierarchy should be organized and named in a way that reflects a strategic view of the organization relevant to the planning of business and information systems, to the extent that this is practical to establish concurrence and a sense of ownership from the business staff involved in planning. Relationship to the existing and/or organizational structure, as well as to the communities of users and other stakeholders outside the enterprise itself, can be handled through mapping to BUSINESS UNITS. See also considerations for enterprise level modeling in the sections on Processes and Events.*
- *In a business function model at the requirements level, functions should be decomposed to a sufficient level of detail to serve as a documented basis for agreement between the business community and the developers that system scope and requirements are clearly understood. Functions outside the planned scope may be included, but should be identified as “manual” or “outside system scope”, though use of the “Automated: Y/N” flag, backed up by text (under “Implementation Notes”). Where a time boxed approach is used, the business function hierarchy must be at a sufficient level of detail that the different functions on the MoSCoW list – showing what “must”, “should”, “could”, or “won’t” be implemented – are clearly distinguished. The function definitions need not include the MoSCoW classification, as long as this is formally kept somewhere and can be easily mapped to the function model. However, where functions that “won’t” be implemented in the function model, they must be identified as outside system scope.*
- *In a system function model, functions must be decomposed to an elementary level at least. Even where a time-boxed approach is used, a complete and accurate system function model should be provided as part of the final system delivery, to support testing, training and ongoing maintenance. Where use case analysis is being used in design, system functions must be decomposed such that each use case is at least at the level of an elementary function (a logical unit of work) if not at a more granular level. An elementary function generally corresponds to a complete, discrete major task that can be accomplished by a user in one sitting. Manual tasks, and activities performed by or using other related systems, may be included in the system function model as long as they are clearly positioned relative to system scope, as already described for the requirements model. They must be included if they are to be referred to in process models, otherwise the decision is at the discretion of the project team, including both the developers and the business users.*

It is expected that the business function model will evolve into the system function model, and not be maintained separately. This may require a realignment of the function hierarchy, where the system design introduces support function definitions that are common to multiple business functions, or

generalizes several business functions so that they actually use the same system function as an implementation point.

Where an enterprise function model exists, the higher level nodes on the business / system function model must clearly align to nodes on the enterprise function model. This may require refinement of the enterprise function model, and/or inclusion of comments in the application-specific business or system function model with respect to this alignment, if it is not obvious.

10. Business Function Definition Standards

10.1. Terminology

A **business function** is an activity that must be performed to meet the objectives of the business, regardless of the method or mechanism by which the activity is carried out. Business functions are defined and diagrammed as a semantic hierarchy, known as the **Function Hierarchy**.

An **elementary business function** is one which, once started, must be completed, or is not considered to have occurred, and leaves no trace of having occurred. It is a 'logical unit of work' in the *business* sense.

An **atomic function** is one that is not further decomposed. An elementary function is not necessarily atomic. This definition is used in Designer.

A *use case* is a type of operation that can be performed on the system by a user, and is described as the sequence of actions, including variants, that yields an observable result of value to the user. It may be useful to think of use cases as "business transactions", more or less equivalent to elementary business functions. The use case formalism is part of the Unified Modeling Language (UML); classes of user interacting with the system in use cases are referred to as *actors* in UML parlance.

10.2. General Standards for Modeling Business Functions

- *For operational functions, a parent function should be completely described by the sum of the functions on the next lower level.*
- *For Decision Support Functions, functions should be decomposed to the point where each elementary function corresponds to a representative data analysis function that can be explicitly mapped to the data it requires. The decision as to what constitutes an effective set of representative elementary functions for DSE requirements within the scope of a parent function needs to be agreed upon between the client community and the analysts documenting the requirements. A function hierarchy need not be used to represent DSE functional requirements for analysis and/or reporting, so long as these requirements can be captured in a formal way that is linked to their use of data.*
- *On the lowest level, in an application system model, functions should easily be classified as manual, clerical, computer, or other mechanisms, but no combinations.*
- *Aim for six to ten subordinate functions per parent function.*
- *An elementary or atomic function is not considered to be fully defined until the following are defined: entity usage; frequency; response; description; and functions triggered.*
- *An elementary function is not considered to be fully defined until the business units that perform the function are defined. Note: the relationship between function and business unit is many to many.*
- *Model generic functions in a separate branch. If there are no special requirements of the generic function do not repeat the function if it is elsewhere in the hierarchy.*
- *Use business rules to simplify the function hierarchy. Look for business rules that are expressed in data terms and define them in the Rules Hierarchy (see Business Rule Definition Standards). This has a beneficial side effect of simplifying the business function hierarchy.*

-
- *Minimize duplication. Avoid sub-trees that seem repetitive unless there are valid business reasons such as:*
 - *There is value in communicating this way to the business users*
 - *There are rules included with the functions that cannot be expressed in data terms.*
 - *Decomposition has gone too far when functions like ‘create entity’, update entity’ and ‘delete entity’ are defined. A function called ‘maintain entity’ is assumed to include the following activities (when they are appropriate):*
 - *create*
 - *record change (update)*
 - *correct*
 - *logically delete*

If one of these activities is not appropriate the range of activities supported should be explicit on the function.

- *When data from more than one area of the logical data model is involved decompose first by data then by functionality. For example:*

Manage Locators and Locations
Manage Locators
 Receive Addresses
 Validate Addresses
 Correct Addresses
Manage Locator Reference Information
 Manage Municipalities
 Manage Postal Codes
 Manage Administrative Regions
Manage Locations
 Receive Locations
 Validate Locations

This is most appropriate for administrative functions.

- *Consider decomposition to be sufficient when one of the following questions can be answered with a “Yes”:*
 - *Does the function map to a module that could be generated?*
 - *Is there no additional complexity beyond what can be captured on the CRUD matrix, through rules related to the data on the CRUD matrix, or on a process model (where process models are being used)?*
- *If you are identifying functions which are doing the same CRUD operations on an entity you are going too far unless the attributes are different in terms of ownership, security, time of update, calculations or derivations involved, etc.*

10.3. Label

Mandatory, 10 characters maximum

A unique label for the function within an application system.

Note: Designer enforces the uniqueness of function labels within an application system, but not between application systems. Naming standards are needed mainly to prevent problems when migrating functions between application systems. Depending on the repository administration policy, migration of functions may not be an issue.

-
- *The function label should, if possible, convey meaning. The function label should not be used to convey the sequence or priority of functions within a branch. (The Sequence in Parent field is used to sequence sibling functions.)*
 - *Do not use spaces or punctuation in function labels.*
 - *Use a two-character label for each main branch at level 1 of the hierarchy.*
Examples
 - RG for the 'registration' branch*
 - CD for the 'card management' branch*
 - EL for the 'eligibility management' branch*
 - ID for the branch containing identification functions*
 - DC for document management functions.*
 - *At level 2, prefix each function label with the two-character identifier of the parent function. Add a single-letter code to label the branch.*
Example:
 - DCA for a branch of document authentication functions in the 'document management' branch of the hierarchy.*
 - *At level 3, use the three-character label of the parent function as a prefix, and add three or four characters to form a meaningful label if possible. This portion of the function label can be formed, in most cases, by a contraction of the main verb used in describing the function.*
Example:
 - DCANTF: Notify interested parties if result of document authentication indicates that a document is fraudulent.*
 - *Below level 3, append numbers to distinguish between functions. Note that these numbers do not imply the sequence in which the functions are performed.*
Examples:
 - DCANTF1 : Communicate fraudulent document to eligibility authorities.*
 - DCANTF2: Notify customer regarding fraudulent document.*

Where design models or specifications are not being captured in Designer, the function label is most likely to serve as an anchor point for references from design to the analysis model. Therefore, the function label should not be changed without ensuring that the effective traceability to and from the design specification is maintained.

10.4. Short Definition

Mandatory, 70 characters

A brief description of the function.

- *The short definition serves as a full name for the function. It is not to be confused with the Function Description.*
- *The short definition should start with a strong verb and indicate the purpose of the function concisely. It should be written in active voice. Keep the short definition brief. It appears on function hierarchy diagrams.*
- *Punctuation marks, underscores and symbols are not to be used in the short definition.*

- *The short definition should specify **what** is done, not **how**. For use cases, it should clearly identify the goal of the use case.*

Examples:

| Poor | Good |
|-------------------------|---------------------------------|
| Fill in form B | Submit request for registration |
| File in brown binder | Archive copy of invoice |
| Call | Communicate / notify / inform |
| Contract is transmitted | Transmit contract to supplier |

10.5. Master Function

The label of the function to which the current function is common. If a master function is entered, the current or copy function assumes all the detail of the master function and no further detail can be entered. A 'common' function, where defined, is identical in every respect to the master function, including data usage.

10.6. Elementary

Indicates whether or not the function is an elementary function. An elementary function is one which, once started, must be completed, or is not considered to have occurred, and leaves no trace of having occurred. It is a 'logical unit of work' in the *business* sense. An elementary function is generally at the level of a use case focused on a specific user goal.

Note: Elementary functions can be decomposed.

- *Elementary functions that will be supported in an application system should all be identified by the end of the Requirements Modeling phase, and must all be identified in the system function model.*

10.7. Frequency

Optional

The frequency with which the function is carried out. The unit of measure is entered in the next field.

- *it is MANDATORY to record frequency for elementary functions.*
- *Use only a positive integer value for frequency.*
- *The unit of time for the frequency field, is usually selected from the following list:
second
minute
hour
day
week
quarter
month
half-year
The default is DAY.*
- *Where none of the terms in the above list is suitable, the unit of time for the frequency field may be a business-specific term, so long as this term is clearly defined in the glossary, and identifies a regular business cycle; e.g. GRADUATION EXAM SESSION.*

10.8. Response Needed

Mandatory

Indicates whether the implementation of the function should be batch or on-line (Overnight or Immediate).

- *As a rule of thumb, use Immediate for functions which are expected to be implemented as interactive transactions; Overnight for functions which are expected to be batched. Operational Reports may be classified as Immediate only where they are small reports run on demand that must be made available immediately to the user*

10.9. Description

Mandatory, 70 characters per line

A brief description of this function.

- *The function description is MANDATORY. It must be recorded even for simple functions that seem adequately described by the short Function Definition.*

The following standards and guidelines relate to the description of business functions:

- *The function description should start with a strong verb and must clearly and fully describe the function.*
- *A function (process) must be described in terms of its essential, rather than its circumstantial, nature.*
- *The function description should not include any element of mechanism, i.e., it should describe what is done, not how it is done.*
- *The function description should include examples of use for clarification.*

The following standards and guidelines relate to the description of use cases as system functions:

- *Use case descriptions may include descriptive text that outlines the nature and specific goal of the function, beyond what is contained in the Short Definition. Beyond that, the use case specification should be entered under standard headings, in the sequence shown and/or described below. Within each of these sections, notes should be numbered. If a particular note applies to all functions under a higher level function, it should be kept at the level of the higher level function.*

PRECONDITIONS:

Record any conditions that must be true before the system function starts, but will not be validated during execution of the function, e.g. the user must be logged on.

MINIMAL GUARANTEES:

For reporting type functions, record the way information will be filtered

SUCCESS GUARANTEES:

For reporting type functions, record the way information will be filtered

ACTOR ROLES:

If the use case involves more participants than the primary actor and the system itself (not necessarily modeled as an actor), then identify the different actors—using the names of BUSINESS UNITS as discussed separately – and the roles in which the actors are known and referred to in the use case text that follows. For instance:

AGENCY FINANCIAL OFFICER is FINANCIAL AUTHORITY

– where AGENCY FINANCIAL OFFICER is identified as a BUSINESS UNIT, and FINANCIAL AUTHORITY appears as an actor role in the scenario description that follows.

MAIN SUCCESS SCENARIO:

This describes a sequence of actions, represented as a script, with each action preceded either by the name of the active role or the word “SYSTEM”, followed by a colon. For instance:

USER: chooses to begin marking a new assessment

SYSTEM: prompts for entry of the student identifier and confirmation of the assessment identifier

USER: keys in the student identifier ...

Since this level of specification is clearly the result of some analysis and design work, it may be appropriate to leave this section out in initial business requirements models, and use the basic description to identify the logical objectives and context of the use case.

10.10. Notes

Optional, 70 characters per line

Any notes or comments about this business function.

- *Function notes should be entered under the following standard headings, in the order shown below:*

INPUT PARAMETERS:

Record any qualifiers for the function

SELECTION PARAMETERS:

For reporting type functions, record the way information will be filtered

SORT ORDER:

For reporting type functions, record the way information will be ordered, and options for the ordering.

PROCESSING REQUIREMENTS:

Record any special features about this function, such as heavy use, or performance criteria, such as required response time.

IMPLEMENTATION NOTES:

Record any details that may affect implementation decisions in this area, or decisions related to the implementation scope or release strategy.

MISCELLANEOUS:

Record general information that will assist in understanding the function, such as historical information or background notes.

CHANGE HISTORY:

Record changes to a function, stating date, reason for change, change made, and initials of the person who made the change. Record changes in chronological sequence.

ANALYSIS QUESTIONS:

Record any questions that need to be answered by the business before the function modeling can be completed.

Each outstanding analysis question should start with A? to identify that it is an open analysis question. When the question is answered it becomes an analysis point, identified by A!

- *Within each of these sections, notes should be numbered. (Other sections may be defined in future.)*

The name of the person who made the note, and the date of the note should be entered in parentheses at the end of each note e.g., (jhabermas jan13/97).

10.11. Function Type

Mandatory for all functions

The type of function. Types available in Designer are:

- Decision Point (to represent a decision point in the Process flow)
 - Report (a simple reporting function)
 - Data Entry (a simple data entry function)
 - Normal (meaning not clear; do not use)
 - Unclassified
- *This field must be filled, although the list of types supplied by Designer is not comprehensive, nor is the meaning of all types clear. Note that Function Type is not intended to indicate an implementation mechanism, but rather to indicate what type of activity the function represents, or what purpose is served by the function*

10.12. Function Calling/Called by

MANDATORY

Although 'Function Calling/Called by' is worded with reference to procedure calls in application program design, it should also be used to represent the situation where one business function 'uses' or encompasses another that is defined separately; e.g. a function known as 'Establish student eligibility' might be depicted as calling another function known as 'Record student school transcript'.

Designer does not include a diagrammer to represent function dependencies or sequences. Although the process diagrammer has some of this information, the function calling network should be explicitly captured as part of the requirements analysis.

The function network can be recorded in Designer using the Trigger tabsheet in the Function Hierarchy Diagrammer. By selecting a function on this tabsheet, a function can be indicated to be executed as a result of executing another function.

If a function, Fa calls function, Fb, then an entry is made in the **Function Triggering Function** screen. Meanwhile, a system **event** (named END-Fa) is created automatically by Designer. These function triggers may not be significant events from a business requirements modeling perspective. Refer to the Event Definition section for more information.

Note: Designer does not check for loops in the calling scheme: Fa calling Fb and Fb calling Fa are accepted at the same time.

11. Business Process Definition Standards

In general, for projects done under ITMB's direction, the only process models required are business context diagrams showing high level functions and flows. These may be done in Visio, with the following additional stipulations:

- The major processes / functions on the diagrams should map directly, by name, to the top layer of functions on the function hierarchy for the same scope.
- Data flows and external organizations or stakeholder groups should be labeled on the diagram.
- The diagram, including its library address on the LAN, should be referenced in Designer.

The following guidelines and standards are for use with the Business Process Modeling tool in Designer. However, any project intending to do extensive process modeling should consult with IMG before determining what tool and standard to use.

11.1. Terminology

A **business process** is a sequence of business functions which are performed in response to a specific business event, to produce a product or deliver a service required in response to that event. The sequence may include alternate branches, representing optional or variant sets of functions depending on conditions identified at specific points in the sequence called *decision points*, where the process flow splits.

Business processes are represented using *process diagrams*.

A process may provide a specific *context* for the performance of a function, including context-dependent rules. For instance, when the Search for a Person is performed in the context of a process to add a new person, a "no match" condition as the result of the search is a positive indicator that the process can continue. In most other contexts in which this function is used, a "no match" condition would be taken as a negative indicator or response. The over-all intent of the process models is to capture those requirements which are specific to the business process or sequence, and allow them to be abstracted from the function definitions.

A **process step** is any one of the functions in the sequence required to execute a process.

A **sub-process** is a business process which is included within a higher level process. A sub-process is recorded as a process step for any higher level process that includes it.

An **event** is an occurrence inside or outside the business that requires an event response to execute. Standards and guidelines for recording events appear in a separate section of this document. The responses required for most major events important to the business will require documentation as processes. Where the response consists of a single elementary function on the function hierarchy diagram, and no process diagram is required, an association must be made between the function in question and the triggering event.

A **process model** is a collection of process diagrams, together with supporting information in the repository, that define how the business responds to the major events within a specific functional scope.

The standards and guidelines in this section relate to a fairly simple set of process diagramming techniques. Other, more complex techniques may also prove useful, and could be regarded as different elaboration on the approach defined here. The most important elaboration are as follows:

- *Data flow diagrams (or DFD's)* identify the flows between functions in a process as containing specific data, and support the mapping of these flows back to views of data in the data model.
- "*Swim lane*" diagrams divide the diagram into horizontal layers to reflect the responsibilities of different user groups in performing specific functions within the process. On such diagrams, the

horizontal stripes representing user groups (or ‘actors’) can be arranged to emphasize the relative proximity of the different user groups to the client, or the source of the triggering event. They also support the capture of process management characteristics, related to the quality and timing of processes.

11.2. Processes within the function hierarchy

It should be emphasized that the Designer repository does not distinguish between a function and a process. Both are stored as functions.

In order to preserve the distinction between the main business function hierarchy, which is intended to be a complete catalogue of all business function, and the process diagrams, which identify the sequence and context for performing functions in response to specific business events, the functions which are expanded into process diagrams will be kept in a separate hierarchy from the main business catalogue of functions.

This hierarchy is called the *process hierarchy*. The root function on this hierarchy will be labeled PROCESSES.

(This arrangement is similar to the separation of data related rules into their own hierarchy).

When functions are added to the PROCESSES hierarchy to become process diagrams, a distinction must be made between the function-as-process and the base business function included in the main function hierarchy. For instance, there could be a process for recording persons, which includes (as a process step) the base function for recording persons, in addition to other functions such as one that searches for duplicate persons. In this example there are two different objects, a process and a function, to which the name “Record Person” might apply. The labeling standards will enforce the distinction between the process and any functions from the main hierarchy which it includes, but the conceptual distinction must also be kept in mind.

The number of processes within an application will likely require an intermediate level of functions one level below the PROCESSES root, which are not themselves processes but are required to cluster the processes into categories. The names of these functions must follow the process naming standards.

11.3. When process models are required as deliverables

A process diagram must be provided when there is more than one business function to perform in response to a specific business event, and where at least one of the following requirements exists for the response (i.e. the process):

- *The elapsed time to completion of the process is to be measured and recorded.*
- *The status of the process will be subject to inquiries and monitoring, to provide client feedback, ensure completion, etc.*
- *Before a new process is initiated, some inquiry needs to be made to make sure this process does not duplicate one that is already going on (e.g. registration of the same client).*
- *There must be controls to ensure that the functions are performed in a specific order, and to create an audit trail that demonstrates that this has happened.*
- *The performance of a function in the context of this process results in different system behaviour from the performance of the same function in a different context (cf. the example of Record Person in the case of newborns, described earlier).*

-
- *Different user groups are involved in performing different functions in the same process.*
 - *The process involves a combination of functions that are expected to be automated with functions that are expected to be manually performed (or performed outside the envisioned system scope).*

If there is doubt as to whether a business process needs to be modeled, and the business representatives believe there is such a need, the process should be modeled, provided that it does not duplicate one that has already been modeled as a response to the same business event. The following conditions may also serve as indicators that a process should be modeled:

- *More than one elementary business function is involved.*
- *Functions from different parts of the function hierarchy are to be performed in sequence.*
- *The function hierarchy does not provide effective information about sequence or processing context, or there is disagreement about how requirements related to context or sequence are to be reflected in the function hierarchy.*

11.4. A suggested approach for modeling process

It should be clear, from the standards and guidelines expressed in this section, that the process model deliverables are to be kept separate from the main function hierarchy, but must use functions from that hierarchy to reflect all requirements except those that are specific to process, i.e. events, flows, and decision points.

In actually developing the process models and FHD's, it may be most useful to begin by identifying major business events and building preliminary process diagrams in response to them, without consideration of any function hierarchy. This is consistent with general approaches to Business Process Re-engineering. The function hierarchy can then be built and elaborated from the contents of these initial process diagrams. A final step would involve going back and rebuilding the process diagrams in the required form, with reference to the function hierarchy.

However, it may be decided that this is not the most effective way to approach the task of preparing deliverables, because an elaborate function hierarchy exists already, because the team has a fundamental disposition towards functional decomposition as a means of analysis, or for some other reason.

11.5. The scope of process representation

The scopes of processes, and the functions represented within them, are assumed to be those portions that are performed within the modeled organization (i.e., in most cases, a Ministry) or directly under its control. If, for purposes of documenting the broader processing context or assumptions, a function beyond this scope is added to a process, this fact must be documented in the function description. An example of such a function might be the search which is usually performed by a pharmacist before recording a new person, as opposed to the search which is inherently part of the process for recording the person.

Similarly, if an additional process diagram is prepared for documenting context, rather than for formally capturing business requirements, this fact must be documented in the description of the function corresponding to the process diagram.

11.6. Functions within the context of processes

At the level represented by a process diagram, a process includes a number of functions. Each of these functions must fall into one of three categories: referenced functions from the main function hierarchy, decision points, and sub-processes.

- *Referenced functions: Implicit in the division of these functions into these three categories is the requirement that any function within a process, which is not either a decision node or a sub-process, must actually reside in the main hierarchy. These functions should be brought into the process diagram using the “Include” Menu Bar command, and the “Global Process” option. The functions must be classed as “elementary” from a business perspective.*
- *Decision points: A decision point involves evaluation of a condition or a range of conditions (analogous to a “case” statement in structured programming), where each possible result is explicitly mapped to a triggering event (equivalent to a control flow arrow on the process diagram, originating in the decision point). A decision point must be fully and uniquely defined as an atomic function in the process diagram where it occurs, and related to its resultant events as well as to any entities from which it requires data in order to evaluate its condition. It must involve no functionality other than this evaluation and triggering. The function type must be “decision point”.*
- *Sub-processes: A sub-process is a process step within a process, which is itself expanded into a number of process steps. A sub-process in one process diagram may be referred to an identical sub-process in a different process diagram using the master function property. Otherwise, a sub-process must be explicitly represented by its own process diagram.*

When these rules are followed, the only functions that appear on a FHD under the PROCESSES root should correspond to process diagrams (including sub-processes) or decision nodes, with the exception of high level functions used to group process diagrams, one tier below the root.

11.7. Process beginnings and ends

Representation of any process which is not a sub-process must include the representation of the triggering event. The name of the event must be distinct from the name of the triggered process, and should include an indication of the agent responsible for the event (e.g. person requests new registration).

The end points of all processes (including sub-processes) must be explicitly identified using outcomes associated with each process step which may end the process. Each outcome must explicitly identify the variant in business response associated with that particular way of ending the process. For instance, a registration process may have alternate branches that result in outcomes such as “client registered and eligible”, “registration request denied”, “client deemed eligible”, etc.

11.8. Labels

Labels on processes are to conform to standards and guidelines similar to those already identified for functions. However, the following qualifications apply to all functions that correspond to process diagrams or to decision points, or to the high level functions used to categorize processes immediately below the PROCESSES root:

- *Use a two-character label followed by a “_P” suffix for each main branch immediately below the “PROCESSES” root. Note that the two character label may not exactly match the two characters in the main function hierarchy, since most processes cut across branches of that hierarchy.*
Examples:

RG_P for the 'registration' branch
CD_P for the 'card management' branch
EL_P for the 'eligibility management' branch
ID_P for the branch containing identification functions
DC_P for document management functions.

For all lower level processes, use the two character prefix followed by a maximum of 7 other characters, including the string "PROC" in the label.

11.9. Short Definition

- Must conform to the standards and guidelines already identified for functions.

11.10. Elementary

To be consistent with a business requirements perspective (as opposed to a technical implementation perspective), the "elementary" property must not be specified on the PROCESSES hierarchy. Functions referenced from the main function hierarchy will inherit the elementary property, where it is relevant, from the functions in that hierarchy.

11.11. Frequency

It is mandatory to specify frequency for processes (not including sub-processes).

- Specifications of this property must conform to the standards and guidelines already identified for functions.

11.12. Response Needed

Mandatory

- *As a rule of thumb, use "Immediate" for processes which are expected to be implemented so as to be completed in a single session.*
- *If a process is expected to run over more than one session, use "Overnight". (Since the process could go on for weeks, the term "overnight" is inappropriate; however, the tool allows no other choices at this point.)*

11.13. Description

Specifications of this property must conform to the standards and guidelines already identified for functions.

11.14. Notes

Specifications of this property must conform to the standards and guidelines already identified for functions.

11.15. Function Type

Mandatory

The type of function: Decision Point or Process, except for the clustering nodes at the top of the hierarchy, which are Unclassified. (Note that the support of "Process" as a Function Type requires an extension to the value set supported by Designer for this property).

12. Event Definition Standards

This section is focused on the modeling of business events, which are the main kinds of events that are likely to be captured as part of requirements modeling in Designer. Business event modeling in Designer, like the process modeling to which it is closely related, is optional.

A business event is an occurrence that causes the business to respond in a planned way. Business events are usually captured in relation to the business processes that they trigger, and are a good starting point for identifying processes, and grouping the functions that relate to each process. Business events as such are to be distinguished from the events used at the lower levels of design specification, to fire rules related to the user interface, data management, or workflow and process integration.

At a more granular level, events can also function as nodes in the network of functions that make up each process, and as trigger points for rules. However, from the perspective of requirements modeling, events at this level should not be specified unless absolutely necessary, as they will tend to distract from the major business events that are of primary importance. The only business events that should be recorded as such, in response to rules depicted in the Rules hierarchy, are Change events that trigger other business functions in the main function hierarchy (see below). Lower level data events are modeled as part of the rule specification in other tool environments, such as GAME.

There are five main types of business events: External, Internal, Temporal, Data Change, and System. These are discussed in subsection 12.3, **Type**.

An event may trigger a business process or function; an Internal, Data Change, or System Event may also be an outcome of a business function.

12.1. General

- *Model business events when the functions are atomic and are part of a significant business process, or when it will not be obvious to the reader how an event is initiated.*

12.2. Event Name

Mandatory, 40 characters

The name of the event.

- *Use self explanatory, descriptive event names.*
- *The name should be composed of a noun indicating the main business object involved in the triggering event, and a verb in the past participle form expressing what happens to or with this object to signal the event.*

Examples:

*Order received
Person deceased
Invoices printed
Application withdrawn*

12.3. Type

Specifies the type of event.

- Temporal events (recorded as *TIME* in Designer) are events that occur at specific, predetermined points in time, i.e. specific dates and/or clock times, usually as part of specific business cycles. The relevant

-
- business cycles may be fixed for the whole business area, e.g. a deadline for budget submissions, or specific to a business relationship, e.g. the expiry date for an agreement or license.
- Data Change events (recorded as *CHANGE* in Designer) are events that result from creation or change of data in ways that signal significant business activity. Typically, in Business Requirement Modeling, these are events that trigger processes, i.e. sequences of separate business functions, and are not containable within the immediate transaction content. (This is an important consideration in distinguishing business events from the events that trigger condition-action rules to control data base integrity and data quality.) Data change events that need to be recognized as business events frequently have to do with threshold values being reached, for instance, expenditures on an account reach a particular point in relation to the budget for the account.
 - External events (recorded as *OTHER* in Designer, and distinguished in the textual description from Internal events) are occurrences outside the business under consideration, which need to be recorded by the business, or requests made by external clients or stakeholder in the business. Respective examples of these are a school enrolment reported to the Ministry of Education, and a request for remarking on a Provincial examination.
 - Internal events (recorded as *OTHER* in Designer, and distinguished in the textual description from External events) are requests made or initiatives taken by people or organizations within the business, for instance, the establishment of a new curriculum standard or contract type, or requests for a review or audit of a particular area of internal or external operations.
 - System events (recorded as *SYSTEM* in Designer) are occurrences that arise within the functioning context of the system itself. They are typically more relevant to the analysis or design stage than to requirements, although Designer automatically creates a system event when a function-triggers-function entry is recorded, and expects them to be defined when the completion of one or more business functions triggers the start of one or more other business functions. Events created in this manner are given names based on the name of the triggering function, for example END-ACCPT. If the name of the triggering function is subsequently changed, the name of the event is not changed.

12.4. On Condition

Optional, 40 characters

- *Do not use this field, use the text description instead.*

12.5. Time Events

12.5.1. Date

Optional, DATE

The date at which the event occurs, if known.

- *Complete this field only for Time events.*
- *Only fully specified dates (DD-MON-YYYY) can be used here.*
- *For more general specifications, such as 30th Sept., any year, or 1st of any month, provide the specification as part of the text description.*

12.5.2. Time

Optional, 5 characters

The time at which the event occurs, if known.

- *Complete this field only for Time events.*

-
- *Use this field to record the time of a daily recurring event.*
 - *Specify the exact time, if known (HH24:MM).*

12.6. Frequency

Optional, 6 digits

For a recurring event, the frequency with which it occurs.

- *Use Frequency and Unit only if there is a regular basis for the event in the scope of that Unit.*
- *Use only positive integer values for Frequency*

12.7. Frequency Unit

The units of measurement for the Frequency.

- *Standard units of frequency are:*
SECOND
MINUTE
HOUR
DAY
WEEK
MONTH
QUARTER
HALF YEAR
YEAR

12.8. Change Events Entity

The name of the entity whose change causes the event (*use for 'CHANGE' events only*).

12.8.1. Attribute

The name of the attribute, if any, whose change causes the event.

12.9. System Description

- *Do not use this field; use Description text instead.*

12.10. Description

Optional, 70 characters per line

A description of this event.

- *Use this field if the event name is not self-explanatory, or to clarify the nature of 'OTHER' events.*
- *Use this field to record any additional conditions that apply to the occurrence of the event; for example, "last Thursday in the month" for Time events that have a frequency of "1 per month".*

12.11. Notes

Optional, 70 characters per line

Any notes or comments about this event

- *Use sparingly.*

13. Rule Definition Standards

Note: the standards and guidelines expressed in this chapter are related to the documentation of a Rule Model in Designer. If some other approved tool, such as GAME, is used to capture rule specifications, the Designer-specific considerations, such as function labeling, do not apply, and tool-specific standards should be used. However, the general approach to rule classification and documentation should still be used.

This section provides standards for documenting business rules in Designer. The approach to the representation and classification of business rules used here is based on several sources, including the approach defined in Oracle's Custom Development Method, the rule classification approach proposed years ago by GUIDE, the rule classification and rule implementation strategies in GAME, and the associations of rules or constraints with different aspects of the metamodel for UML.

13.1. General Approach to Modeling Business Rules

Many rules are naturally expressed in the model without the need for additional specification, because their specification is naturally captured in other structures and properties in the model. For example, there are "optionality" properties on both relationships and attributes. However, such a rule must apply to every instance of the relationship or attribute in question. Where a rule of this kind is unique to a subtype, or subject to a particular condition (i.e. "mandatory if ..."), then a separate rule specification is required.

The approach proposed for modeling business rules assumes that the data required to support business rules will be represented on the data model, even for those data-related rules that are not explicitly captured through model notations and model properties. Since Designer does not provide any support specifically for rules, the rules themselves are to be recorded as business 'functions'. With the exception of a few types of rules, which are directly associated with business or system functions, the 'rule functions' are recorded in a separate function hierarchy. (The rule types that may be associated directly with functions, such as function authorization rules, are explicitly identified in the detailed discussion of rule types, below.) Each 'rule function' is to be linked to the data model through the data usage (CRUD) matrix.

This approach provides:

- a consistent method for documenting 'unstructured' business rules, and access to that documentation through standard Designer reports and matrix diagrams,
- a basis on which these rules, as requirements, can be traced back from application design specifications,
- full impact analysis capability through the CRUD matrix,
- a basis for further work in developing a more structured rules model, as methods for rules modeling evolve.

13.2. Rule classification

There are six main classes of business rules to be handled here:

- **Data integrity and Quality rules** control or monitor the structure and content of the data in the data base. For the most part they are "static" rules, which define the states in which data may exist. They affect the operations that insert or change (or, in some cases, delete) data, but are defined purely in terms of the data itself, and without reference to these data operations (other than through the CRUD mappings).

- **Value Transition rules** govern the change from one data content to another. They can be thought of as state transition rules, where specific attribute values indicate the state of the entity in question, according to a specific classification perspective.
- **Operational integrity rules** make sure that specific functions / operations are performed when they should be, and are not performed when they should not be, based on process flow definitions, event definitions and constraints, and triggering from other functions and/or data.
- **Business control rules** identify specific fiscal constraints, such as authorization, audit logging, and balance control.
- **Information Integrity Rules** govern the production of aggregate and derived information products from the data base, including derived data calculations, and aggregations and transformations of data structures. They are intended to be used primarily in the specification of requirements for Decision Support systems. The descriptions in this version of the document are provisional.
- **Function detail rules** are business rules specific to the decision process embodied in a single, complex, rule intensive business function. They should be specified in relation to the function itself, and not as part of the RULES hierarchy.

A few general observations need to be made about this classification, and its sub-classifications:

1. The rule definitions start with a very strong orientation towards the data model, and persistent data. The first major classification, Data Integrity and Quality rules, is broken down into groups corresponding to the major constructs in a data model – entities, relationships, domains, and attributes – and really consists of extensions to the static definition of persistent data structures.
2. The second major classification, Value Transition rules, has to do with dynamic change, but is still expressed entirely in terms of the data structure.
3. The next two major groups, Operational Integrity Rules and Business Control Rules, start to broaden the focus and include some rules that can be defined entirely in the context of specific functions, but the sub-classifications, in each case, begin with rules that are expressed entirely or primarily in terms of data.
4. The last two major groups, Information Integrity Rules and Function Detail Rules, are focused primarily on types of application systems different from conventional transaction processing. Information Integrity Rules are conceived primarily for the context of Decision Support systems, and Function detail rules are conceived primarily for the context of rule-intensive business functions. However, aspects of both these kinds of systems may be present in more conventional transaction processing systems, so these extended rule types may be pertinent in other contexts, and should not be ignored.

Other major classes of rules may be added in future.

The following table shows a detailed breakdown of the different rule types and the prefixes to be used in naming the rule functions for each detailed type. More detailed descriptions and specific guidelines for the various rule types can be found in later subsections of this chapter, as indicated by the section numbers in parentheses.

| Rule Class | Rule Sub-Class | Rule Type | Description | Function Prefix |
|---|-----------------------------|----------------------------|---|-----------------|
| Data Integrity and Quality Rules (13.4) | Entity level rules (13.4.1) | Unique ID rules (13.4.1.1) | A Unique Identifier rule defines which (combination of) attribute(s) and / or relationship(s) can be used to identify an occurrence of an entity. | UID |

| | | | | |
|------------------------------------|-------------------------------|--|---|----------------------------------|
| | | Single Entity rules (13.4.1.2) | Any other integrity rule specific to an occurrence of a single entity type, independent of its relationships or attributes. | ENT |
| | Relationship Rules (13.4.2) | Restricted Relationship Rule (13.4.2.1) | Any rule governing the existence and/or management of relationships between entity instances. | RER |
| | Domain Rules (13.4.3) | Algorithmic Domain rule (13.4.3.1) | Any algorithmic rule constraining values in a domain, e.g. an edit mask, a check-digit algorithm, a series of domain ranges. | DOM |
| | Attribute Rules (13.4.4) | Mandatory attribute (13.4.4.1) | A rule that ensures that required attributes exist and are assigned values as necessary. (A rule specification may be required for attributes of supertypes constrained specifically for subtypes.) | MND |
| | | Local attribute constraint (13.4.4.2) | An edit cross-check, where all the attributes involved are in the same entity instance. | TUP (“Tuple” rule in CDM) |
| | | Extended attribute constraint (13.4.4.3) | An edit cross-check, where at least some of the attributes involved are in different entity types and/or instances. | INT (“Inter-entity” rule in CDM) |
| Value transition rules (13.5) | Value Transition (13.5) | Value Transition (13.5) | A rule constraining the possibility of change of an attribute from one value to another, including changes to or from null values, or changes implicit in the creation or deletion of the entity. | TRS |
| Operational integrity rules (13.6) | Operation constraint (13.6.1) | Create constraint (13.6.1.1) | A constraint specific to the creation of an entity instance in the data base. | CRE |
| | | Update constraint (13.6.1.2) | A constraint specific to the update of an entity instance in the data base. | UPD |
| | | Delete constraint (13.6.1.3) | A constraint specific to the deletion of an entity instance in the data base. | DEL |
| | | Modify constraint (13.6.1.4) | A constraint specific to the update or deletion of an entity instance in the data base. | MOD |
| | | Other Operation constraint (13.6.1.5) | A rule to prevent an operation on an entity / object type from occurring under specific conditions | OP |

| | | | | |
|-------------------------------|------------------------------|--|---|-----|
| | | Multiple Operation constraint (13.6.1.6) | A rule to prevent any of several operations on an entity / object type from occurring under specific conditions | MOP |
| | Action trigger (13.6.2) | Action trigger | A rule to ensure a specific action is taken in response to a specific operation / event and condition. | TRG |
| | Process constraint (13.6.3) | Schedule constraint (13.6.3.1) | A rule to determine the calendar or clock time, elapsed time, or other constraint on the scheduling of a specific function or process in relation to the triggering of an event or process. | N/a |
| | | Other Process constraint (13.6.3.2) | A rule to constrain or direct the flow of control within a defined business process, or the initiation or completion of such a process. | N/a |
| Business control rules (13.7) | Audit Logging Rules (13.7.1) | Audit Logging Rules | A rule enforcing the logging of the previous version of an entire entity (i.e. database row) or an attribute (i.e. a column), when it changes. | LOG |
| | Authorization Rules (13.7.2) | Data access authorization (13.7.2.1) | A constraint limiting the authority of a user in a specific role or business context to perform a specific data access, including read as well as write access. | AUD |
| | | Function Authorization (13.7.2.2) | Function-specific authorizations can be handled by mapping the functions in the main function hierarchy to Business Units, using the Designer Matrix Diagrammer (BunFun matrix). | n/a |
| | | Other Authorization (13.7.2.3) | There may be other business or system resources, such as specialized printers or forms, or message queues within the system context, which require authorization. | AUO |

| | | | | |
|------------------------------------|------------------------------------|---------------------------|---|-----|
| | Balance Control (13.7.3) | Balance Control | A rule constraining / ensuring the collective integrity of data stores and/or processes, through cross-checks on aggregate values such as sums and tallies. | BAL |
| Information Integrity Rules (13.8) | Derivation Rules (13.8.1) | Derivation Rules | Derivation rules provide the algorithms used in deriving a value from the value or values of other attributes, constants, and/or dynamic parameters. | DRV |
| | Data Matching Rules (13.8.2) | Data Matching Rules | Data matching rules provide the algorithms or logic used to determine if two objects relate to the same real world thing – e.g. the same person, or organization, or contract. | DMA |
| | Data Transformation Rules (13.8.3) | Data Transformation Rules | Data transformation rules identify the ways in which specific data structures are transformed into other data structures, in support of specific conversions or interfaces. | DTR |
| Function Detail Rules (13.9) | Function Detail Rules | Function Detail Rules | These are business rules specific to the decision process embodied in a single, complex, rule intensive business function. They should be specified in relation to the function itself, and not as part of the RULES hierarchy. | n/a |

13.3. General Standards and Guidelines for Recording Rules as Functions

13.3.1. Rules Hierarchy

For the purpose of documenting ‘business rules’ which cannot be defined directly on the entity-relationship model, a separate function hierarchy must be created.

- *Create a dedicated hierarchy for business rules that are recorded as functions, with branches for each class of business rule. Note that there are some exceptions where rules of specific types are permitted in the main function hierarchy; see for instance 13.7.2.2 Function Authorization Rules and 13.9 Function Detail Rules.*
- *The root function on this hierarchy must be labeled RULES, or, alternatively, BR.*

-
- *The label on each function in the hierarchy must begin BR, followed by the three character acronym for the rule type (e.g. 'BRUID').*

13.3.2. Rule descriptions and specifications

- *The description of each function should express the entire rule, including constraints, even though some aspects of this description may be redundant with the rule name, the CRUD, etc.*
- *Many rules, and integrity rules in particular, will include a condition expression which must / should be true in order for the operation to proceed, the data to be stored, etc. Either the word "must" or the word "should" must be used as part of the text of such a rule specification. If "must", the rule is a constraint and will inhibit the operation. If "should", the rule is to be used for validation or monitoring – usually of data quality – and to generate a warning message and/or quality statistic entry.*
- *The applicability of rules may vary between subtypes and supertypes. It is essential that the both the text of the rule and the CRUD are referring to the appropriate level of typing. It is also essential that rules on subtypes are more constraining than rules on the supertype, if overlapping rules are defined; i.e. what is permitted on the supertype may be restricted on the subtype, but not the other way around.*

13.3.3. Mapping of Rule Functions to the Data Model (CRUD Matrix)

- *The relationship to data that may trigger each rule, or be otherwise implicated by the rule, must be represented through the CRUD matrix. Specific guidelines for the use of the CRUD are included with the discussions of the various rule types below. Note that the CRUD mapping may vary between entity and attribute level, according to the kind of rule involved.*

Specific guidelines for documenting each rule type, together with other more detailed discussion of the rule types, appear on the next few pages.

13.4. Data Integrity and Quality rules

Data integrity rules control or monitor the structure and content of the data in the data base. For the most part they are "static" rules, which define the states in which data may exist. They affect the operations that insert or change (or, in some cases, delete) data, but are defined purely in terms of the data itself, and without reference to these data operations.

Data integrity rules may or may not be enforced. Where they are to be enforced, i.e. where data base operations are to be inhibited if the resulting data state breaks the rule, then the rule must contain the word "must".

Alternatively, the rule may be used to monitor the state of data, with a message or other record being written if a rule infraction is detected. In such a case, a data integrity rule becomes a data quality rule, and must be written with the word "should" instead of "must".

Data integrity and quality rules are classified as to whether their focus is primarily on entities, relationships, domains, or attributes. One way of thinking about these rules is as extensions to the specific standards for modeling these various aspects of data models, as described in earlier chapters of this document.

13.4.1. Entity level rules

Entity level rules represent extensions to the formal definition of entities, as outlined in chapter 4 **Entity Definition Standards**.

13.4.1.1. Unique Identifier rules

A Unique Identifier rule defines which (combination of) attribute(s) and relationship(s) can be used to identify an occurrence of the entity.

Although native Designer metadata supports some specifications of this control, it does not support situations where the unique ID is established through relationship to another entity (other than the parent), or where Unique Identifier constraints vary among subtypes. The rule goes beyond UID specifications in the Designer metadata, because it allows for ID uniqueness within a subtype, and also for unique reference from an ID stored in a separate entity. It also allows for more than one Unique ID constraint on the same entity type.

Example:

An employee must be uniquely identified by an employee number. (This example assumes that, as in our Enterprise Data Models, a BUSINESS IDENTIFIER such as employee number is in a separate entity from the PARTY or PARTICIPANT it identifies, in this case the employee.)

- *If the Unique identifier rule applies to every occurrence of an entity type, use the Unique Identifier object in Designer to record this type of rule, as described in section 4.9 Unique identifier.*
- *Where unique identifier rules vary for different subtypes, or involve relationships to entity types other than the parent, record them as functions in the RULES hierarchy, using UID as the prefix for the function label.*
- *A UID rule function should have CRUD mappings to the insert—and if applicable the update – of the entity type in question. Where attributes of other entity types are used to form an ID, these attributes should also have CRUD mappings as references ('R'). There should also be insert and update mappings to any intervening entity types used to link the subject entity to the entity containing the unique ID, or any portion of it.*
- *There may be more than one unique identifier for an entity. In such a case, if and as necessary, unique identifiers may be prioritized by sequence numbers.*
- *Note that, when the identifier is in a different entity type from the identified target (e.g. a BUSINESS IDENTIFIER such as employee number qualifying a PARTY or PARTICIPANT such as employee, as in the example), the rule is not always or only ensuring the uniqueness of the target. In addition or instead, it may be ensuring the uniqueness of the mapping from the identifier to the target (e.g. that no employee number can refer simultaneously to two or more employees).*
- *In some cases, particularly when all or part of the unique identifier is in another entity type, the rule will include time dependency; i.e. the identifier must be unique within an effective time period, often identified as the effective time period of the identified entity, or of an association that links the identifier to the entity. In such an instance, the time dependent nature of the rule must be made explicit. This can be done either by stating the time dependency explicitly in the rule description, or by including the effective start date as part of the unique identifier.*
- *Should there be a requirement to specify a Unique Identifier rule as being conditional on something other than the effective date, such as the value of some other attribute or the existence of some relationship, it would be preferable if this condition could be associated with a subtype. However, if necessary, the condition can be expressed as part of the rule description.*

13.4.1.2. Single Entity Rule

A Single Entity Rule is any other integrity rule specific to an occurrence of a single entity type, independent of its relationships or attributes.

Example:

There must be no more than 20 departments.

- *Record Single Entity Rules as functions. Use ENT as the prefix for the function label.*
- *An ENT rule function should have CRUD mappings to the insert—and if applicable the update – of the entity type in question. If other occurrences of the same entity type are used in rule evaluation, there should also be a reference ('R') mapping to this same entity type.*

13.4.2. Relationship Rules

Relationship rules govern the existence and/or management of relationships between entity instances. Many relationship rules, such as optionality, cardinality, and transferability, are specified as properties of the relationship itself, as described in chapter 5 **Relationship Definition Standards**. Where this level of specification will not work, because the rule is conditional for some instances of a relationship in the model, or varies for specific subtypes, the rule must be recorded as a Restricted Relationship Rule.

Several cautions must be kept in mind with respect to the coding of Relationship Rules:

There are guidelines provided for defining redundant relationships involving entity subtypes and supertypes, where the relationship properties vary. These guidelines are described in section 5.6 **Relationships among subtypes**. They may be used to cover the subtype specific rules, instead of adding functions to the RULES hierarchy.

Rules on a subtype must always be at least as constraining as the rules on a supertype. For instance, a relationship may be optional for the supertype and mandatory for the subtype, but not the other way around.

“Relational integrity” rules, which identify the action to be taken when a delete or update request is received for a parent entity, are not covered in this section. They are actually Operational Integrity rules; see section 13.6.

13.4.2.1. Restricted Relationship Rules

Where a rule is applied conditionally to some instances of a relationship, or to specific subtypes of one of the participating entities, a Restricted Relationship rule should be specified.

Example:

An employee can be managed only by an employee with job 'MANAGER'.

- *Restricted Relationship rules are recorded as functions; use the prefix RER for the function label.*
- *An RER rule function should have CRUD mappings to the insert—and if applicable the update – of the entity type that holds the foreign key in the relationship. This is the entity type on the 'many' end of the relationship in question (assuming the model is expressed such that all many-to-many relationships are resolved by associative entities). If attributes are referred to in the rule, there should also be reference ('R') mappings to these attributes.*

13.4.3. Domain Rules

Domain rules govern the use of Domains, or generic definitions of the allowable formats and sets of values for attributes, as outlined in chapter 6, **Domain Definition Standards**. They may be regarded as extensions of the specifications discussed in that chapter.

Because domains do not imply an explicit data context, as entities, relationships, and attributes do, it is relatively rare that functions in the RULES hierarchy can be used to extend domain definitions. However, any rule specific to an attribute should be considered, to make sure it would not be more appropriately specified at the level of the domain.

- *Format, maximum length and precision are recorded as properties of the domain.*
- *Enumerated Value sets for Domains are specified through the domain definition.*
- *Upper and lower bounds for domains consisting of a single range of values are specified through the domain definition.*

13.4.3.1. Algorithmic Domain Rule

This type of rule governs any domain whose values have to be validated by an algorithm, or some means that cannot be specified as part of the domain definition.

Examples:

Employee salary must be a multiple of 1000.

Client number must be 10 digits, with the 10th a check digit calculated on the basis of the other 9.

- *Record any Algorithmic Domain rule as a function, using DOM as a prefix for the function label.*

13.4.4. Attribute Rules

Most rules and constraints related to attributes in themselves are defined as properties of the attribute definition, or of the domain associated with the attribute. See chapters 6, **Domain Definition Standards**, and 7, **Attribute Definition Standards**, and also the section on Domain rules immediately above.

- *Attribute format, maximum length and precision are recorded as properties of the domain with which the attribute is associated.*
- *Allowable values are defined as part of the domain definition, as an enumerated list, as a minimum value, as a range, or as another rule defined at the domain level. Allowable values should NOT be defined as properties of the attribute itself.*

13.4.4.1. Mandatory Attribute rules

A mandatory attribute rule ensures that required attributes exist and are assigned values as necessary. In general, if an attribute is always mandatory on an entity type, this requirement can be indicated by specifying “Mandatory” for the Optionality property of the attribute (see 7.7 **Attribute Optionality**). A mandatory attribute rule is used to identify an attribute as being mandatory under specific circumstances, e.g. when another attribute is present or has a specific value. A rule specification may also be required for attributes of supertypes constrained specifically for subtypes.

e.g. Nationality is mandatory for a student who is not eligible for funding as a resident of British Columbia.

- *Record any Mandatory Attribute rule as a function, using MND as a prefix for the function label.*
- *For this rule type, there should be a CRUD mapping for the Insert and update of the attribute in question, and a reference ('R') to any other entity that needs to be involved in evaluating an associated condition.*

13.4.4.2. Local attribute constraint

A Local Attribute rule defines interdependent allowable attribute values within the same entity occurrence (tuple). A Local Attribute rule can be implemented as a check constraint. Example:

Employee termination date must be later than employee hire date.

- *Record Local Attribute rules as functions. Use TUP as the prefix for the function label.*
- *For this rule type, there should be a CRUD mapping for the Insert and Update of all the attributes involved in the rule. A CRUD mapping for Delete operations may also be required.*

13.4.4.3. Extended attribute constraint

Extended Attribute rules define attribute values that are allowed to be assigned depending on the values of attributes of *other entities*.

Example:

The sum of all outstanding orders for a customer should never exceed the credit limit of that customer.

- *Record Static Extended Attribute Rules as functions, using the prefix INT as the prefix for the function label.*
- *For this rule type, there should be a CRUD mapping for the Insert and update of all the attributes involved in the rule. A CRUD mapping for Delete operations may also be required.*

13.5. Value Transition Rules

Value Transition rules define allowable attribute values based on the previous value of the same attribute. In addition to constraining the possibility of change from one value to another, the rule may constrain changes to or from null values, or changes implicit in the creation or deletion of the entity.

Example:

Allowed transitions for marital status of a person are:

- unmarried to married
 - married to divorced
 - divorced to married
 - married to widowed
 - widowed to married.
- *Value Transition Rules may be recorded as functions, using the prefix TRS as the prefix for the function label.*
 - *For this rule type, there should be a CRUD mapping for the update of the attribute governed by the transition rules, and a Reference ('R') mapping for any other entity type involved in conditions applying to the transition rule. If the rule includes constraints on insertion and deletion as well, there should also be CRUD mappings for these operations on the governed attribute.*

-
- *All transition constraints for a single attribute may be included in the same Rule Function, or optionally, each may be recorded as its own function.*
 - *If, at analysis time, the transition rules are captured as metadata in some other approved tool that is integrated with the data models and system configuration management environment, such as GAME, it is unnecessary to specify them in the rules hierarchy as well.*

13.6. Operational Integrity Rules

Operational integrity rules make sure that specific functions / operations are performed when they should be, and are not performed when they should not be, based on process flow definitions, event definitions and constraints, and on triggering from other functions and/or data state changes.

Note that we are using the term “operation” in the sense in which it can be used in object oriented analysis and design, and is used in UML. It is not limited to the Create, Update, and Delete operations that are recognized by the DBMS and can be governed by triggers at that level. An operation is a formally defined service that can be requested in relation to an object. Thinking of entities as persistent data objects, we can expect that some or all of the CRUD operations (Create, Refer, Update, and Delete) can be supported for any entity. However, a full model of persistent data behaviour might also include others, either generic (such as Validate, Print, Render in XML, Archive) or business specific (such as Schedule, Ship, Pay, Evaluate).

(Defining such a model in Designer is challenging, in relation to both the use of function hierarchy and the specification of the CRUD. Operations other than the data base specific ones should probably be defined in a separate hierarchy, with a node for each object / type level for which operations are defined, and then cross-referenced to functions that use them. Each operation will then have its own CRUD mapping to the data base specific operations it requires. Use of these extended operations by other functions must then be represented through function-to-function triggers. This approach is not generally recommended for most use of Designer; however, it may be appropriate in some contexts, and is certainly the direction that functional modeling is likely to take in future, and is already taking with UML. For these reasons, this rule classification approach encompasses this broader scope of operation definition and use.)

There are several other possible areas of confusion with regard to the classification of rules that might affect operational integrity. In the classification we are using here, State transitions, which could be considered operational integrity rules (and operation constraints in particular) have been classified separately; see 13.5 **Value Transition Rules**. However, Relational Integrity Rules, which might at first appear to be data integrity rules (see 13.4) are actually classified as Operational Integrity Rules, because they control behaviour and do not simply describe static data states. They are either Operation Constraints or Action Triggers, depending respectively on whether the rule inhibits or propagates deletes and updates from a parent entity to any existing children.

13.6.1. Operation Constraints

An Operation Constraint prevents a specific operation or function from taking place under specific conditions. (See the discussion of the term “operation”, in the preceding section.)

Data Operation constraints (create, update, delete and modify constraints) define conditions under which –

- an entity occurrence can be created or deleted;
- an attribute value can be inserted, updated, or cleared;
- a relationship can be created, deleted, or transferred.

Operation constraints can also be considered as including rules that inhibit any given function or operation from taking place. Though these are best specified in direct relation to the specific function they effect on the function hierarchy, they can be considered rule functions, and are described below in section 13.6.1.5 **Other Operation Constraints**.

There is a separate category for rules that describe what will happen after a change in the data (creation, update or deletion of an entity or attribute) has taken place. See section 13.6.2 **Action Triggers**.

13.6.1.1. Create Constraints

A Create Constraint restricts the creation of an entity or relationship.

Example:

An item cannot be added to an order that is already approved.

- *Record Create Constraints as functions, using the prefix CRE as a prefix for the function label.*
- *Create constraints must have CRUD mappings for creation of the entity type in question, and reference mappings to any attributes involved in the condition evaluation.*

13.6.1.2. Update Constraints

An Update Constraint defines conditions for the update of an attribute, entity or relationship.

Some constraints on updates are defined in terms of the transferability property of a relationship: a foreign key can be updated only if the relationship is transferable. For instance, a student enrolment cannot be transferred from one school to another, so we would have the relationship from ENROMENT to SCHOOL identified as “not transferable”. Note that this can be done only if the rule applies to all instances of the relationship in question. At the level of the Analysis model, it is also restricted to relationships involving entity types that are materialized as tables..

- *All other update constraints must be recorded as functions in the RULES hierarchy, using UPD as the prefix for the function definition.*
- *Update constraints must have CRUD mappings for update of the entity type in question, and reference mappings to any attributes involved in the condition evaluation.*

13.6.1.3. Delete Constraints

Delete Constraints define conditions for the deletion of an entity or a relationship.

A particularly important set of delete constraints are the rules defining the conditions under which a parent entity can be deleted, and what should happen to child entities if the parent is deleted. A delete constraint can be used to restrict the deletion of a parent entity to situations where there are no dependent children. For instance,. *A department cannot be deleted when employees exist who are working for that department.*

There are two important points to recognize about delete constraints on parent entities:

- If the constraint applies to all instances of the parent, it can be implemented as a referential integrity rule in the data base design. This is the database design default; therefore, such rules do not have to be defined in the Rules Hierarchy.
- If a parent entity can be deleted when it has children, then there must be a provision for what happens to the children. This provision should be recorded separately as an Action Trigger, not a Delete constraint. The action can be implemented directly as part of the database design if it applies to all instances of this parent’s children, and indicates either –

-
- that the deletion is “cascaded” to delete the children when the parent is deleted (e.g. all order lines are deleted when an order is deleted); or
 - that the children are not deleted but the relationship is nullified when the parent is deleted (e.g. when a department is deleted, the employees remain in the database but are not assigned to any department).

Examples of other Delete constraints:

A file may be deleted only when the end date of that file is in the past.

A client may be deleted only when there are no receivables against that client’s account.

- *Delete constraints must be recorded as functions in the RULES hierarchy, using the prefix DEL to label these functions.*
- *Delete constraints must have CRUD mappings for deletion of the entity type in question, and reference mappings to any attributes involved in the condition evaluation.*

13.6.1.4. Modify Constraints

A Modify Constraint is a rule that combines the effect of an update constraint and a delete constraint. It is equivalent to the specification of two rules, both having the same condition.

- *Modify constraints must be recorded as functions in the RULES hierarchy, using MOD as the prefix for the function definition.*
- *Modify constraints must have CRUD mappings for both update and deletion of the entity type in question, and reference mappings to any attributes involved in the condition evaluation.*

13.6.1.5. Other Operation Constraints

Other constraints on operations or functions may be documented as rules. (For a brief consideration of how operations could be reflected in Designer, see the introductory discussion in 13.6 **Operational Integrity Rules**.)

- *In general, rules that constrain the execution of functions or operations in the main function hierarchy will be expressed as sub-functions directly associated with the functions they constrain, unless they can be expressed as rule types already identified above, or as data or function authorization rules.*
- *Should there be a need to include other operation constraints directly in the Rules hierarchy, they may be recorded using OP as the prefix for the function definition, and creating a function trigger from the operation to the rule. CRUD mapping should include a reference (‘R’) to any attributes / entities involved in evaluating the constraint conditions.*

13.6.1.6. Multiple Operation Constraints

In some cases, the same rule may be used to constrain more than one operation. In such a case, the rule may be considered a Multiple Operation Constraint. If one of the operations is a data base operation and the other is not, or if one of the operations is a data base create, it must be defined in this way. (Note that if the rule is constraining only updates and deletes, a Modify Constraint (MOD) may be used).

- *Any rule that constrains more than one operation type should be included in the Rules hierarchy, using MOP as the prefix for the function definition. Create a function trigger to the rule from any operation or function it constrains directly. CRUD mapping should include a reference (‘R’) to any attributes / entities*

involved in evaluating the constraint conditions, and the appropriate operations 'C', 'U' or 'D' for any data base operation it constrains.

- *Note that it is unnecessary to use a rule to constrain a function or operation that uses a data base operation, if the data base operation is already constrained by the same rule.*

13.6.2. Action Triggers

An Action Trigger determines an action to be taken when a specific event or condition occurs. This is usually an automated, derived (secondary) action that takes place after a data state change. The change event defines which change of data triggers the action. This can be one, or a combination, of the following events:

- Creation, update, or deletion of an entity
- Creation or update of an attribute value
- Creation, transfer, or deletion of an entity relationship.

Examples:

When the end date of a customer of a telephone company is set, create a work order of type 'Disconnect customer'.

When fulfillment of an order causes the quantity on hand of any item in the warehouse to fall below the order threshold, issue an order to the supplier for the standard order quantity.

If a new movie is added to the Titles inventory, print a new movie catalog.

Note that the trigger may be governed by conditional logic, as well as the triggering event.

It is possible, at least conceptually, to have an action triggered by an event other than a data state change; however, it is unlikely that this will be necessary with the current use of Designer. Other kinds of events can be explicitly modeled as Events (see 12 **Event Definition Standards**), and associated with the resultant functions or processes. It is still possible that an event might be associated with a persistent object operation that is not a basic data base operation; but, since Designer does not support the modeling of objects and their behaviour in this way, the operation would have to be represented as a function in any case.

- *Any conditional triggering directly associated with the logic of functions can be represented directly in the main function hierarchy, either as part of the description of the function in question, or through a rule function that is specified separately but represented as a sub-function of the function it qualifies.*
- *Where action triggers are based on data base events, record them as functions in the RULES hierarchy, using the prefix TRG as the prefix in the function label.*
- *Action Triggers in The RULES hierarchy should have CRUD mapping to the data base event(s) that may cause them, and reference ('R') mapping to any attributes used in conditional logic.*

13.6.3. Process Constraints

13.6.3.1. Schedule Constraints

A Schedule Constraint may determine the calendar or clock time, elapsed time, or other constraint on the scheduling of a specific function or process in addition to other functions, rules, or events that may trigger it.

Examples:

A securities transaction cannot be posted after the close of the market.

Terms of a proposed contract are invalid after 30 days if the contract has not been finalized.

A Schedule Constraint may be related to a Temporal Event, and almost always has a direct bearing on the scheduling of a specific function or process. It may be possible to represent it explicitly as a constraint on a process model. Otherwise, it can be represented directly in the main function hierarchy, either as part of the description of the function in question, or through a rule function that is specified separately but represented as a sub-function of the function it qualifies.

13.6.3.2. Other Process Constraints

Other rules besides Scheduling Constraints may be required to constrain or direct the flow of control within a defined business process, or the initiation or completion of such a process. Such rules are more usually, and perhaps more usefully, represented as structures and characteristics on a process model, but may need to be defined separately under specific circumstances. In such circumstances, any other constraint on a process may be represented as a sub-function qualifying that process, as already described for Schedule Constraints.

Note that Balance Control rules, which are often used as process constraints, on batched financial processes in particular, are covered separately in section 13.7.3 **Balance Control Rules**.

13.7. Business Control Constraints

Business Control constraints are rules that enforce specific instances of general business policies related to auditing and security. They apply to three distinct kinds of policy: Audit Logging, Access Authorization, and Balance Control.

13.7.1. Audit Logging Rules

Audit logging rules are used to force the recording of data that is changed or subject to changes, so that there is an audit trail or history of all changes to the data. (Note that this is in addition to the widespread practice of including time stamps and user IDs on all data base records, to indicate when they were created and when last updated, and by whom.) Audit logging can be done at the entity (table) or attribute (column) level.

Logging rules need to be little more than indicators that logged instances are to be kept, with an indication (by name) of where the logging takes place, and also when. For attribute logging in particular, it is probably a good idea to add the new value of the attribute to the log each time it is changed, so that the log history is complete, and always includes the current value – even though, for audit purposes, it is not necessary to make a separate copy of each value until just before it is replaced.

(Note that audit logging is different from database logging. Data base logging is used to ensure that the data base can be recovered or rebuilt if necessary, and is optimized for that purpose. Data base logs are not generally usable to provide business information directly on line.)

- *Audit logging rules may be recorded as functions in the RULES hierarchy, using LOG as the prefix in the function label.*
- *Audit Logging Rules in the RULES hierarchy should have CRUD mapping to the data base operation(s) that may trigger them. Mapping should be at the entity or attribute level to correspond with the level of data logging required (entity/ row or attribute / column).*

13.7.2. Authorization Rules

Authorization rules permit or prevent users of a system from using specific data, functions, or other identified resources of the system.

13.7.2.1. Data Access Authorization Rules

Data Access Authorization rules govern the use of data by users of a system. The levels of permission usually vary as to the kind of operation performed on the data, and relate to the users' roles; so that a user in one role can only read certain data, another can read and update it, another can create new instances but not delete them, etc.

Whether or not a specific rule function is required will depend on the complexity of the authorization:

- If an authorization can be expressed as a mapping between a role type (recorded as a Designer Business Unit) and an entity or attribute type, then it can be represented using the Designer matrix diagrammer. This is commonly called *vertical authorization*, as it holds for all rows in the data base. (e.g. Only the managers in the Comptroller's office can create new instances of the Ledger Account entity.)
- If the authorization is dependent on the actual content of the data, as well as its type, then a rule must be specified. The rule may also be dependent on data about the actual user (e.g. An invoice can only be approved by an employee who works in the department which issued the invoice). This is commonly called *horizontal authorization* as it applies to only certain rows or instances of the data at any given time.
 - *Horizontal data authorization rules should be expressed as functions in the RULES hierarchy, with prefix AUD. They should have CRUD mappings to reflect the types of access on the data being controlled, as well as Reference ('R') mappings for any attributes used to evaluate the conditions for access.*
 - *Data access authorization rules against subtypes may also be expressed as functions in the RULES hierarchy. This may be a particularly useful approach if a supertype is implemented as a single table, since that effectively makes any authorization rules specific to its subtypes 'horizontal'.*
 - *Since this set of standards and guidelines relates to the documentation of requirements and analysis, the rule classification should not be driven by the way in which a rule is supposed to be implemented. Data Access Authorization rules may be used to document a requirement to limit the range of data a user may see as well as the data that may be updated, and so could be the basis for implementing a virtual private database, rather than rules that are triggered in the application logic.*

13.7.2.2. Function Authorization Rules

Function authorization rules govern the use of business functions by business users (e.g. Only regional managers have the authority to approve payment of an invoice). If function authorization rules are simple, they can be expressed entirely by mappings between Business Functions and Business Units, using the Designer Matrix Diagrammer (BunFun matrix). If a function authorization rule is conditional, the conditional logic must be expressed in relation to the function being governed. The rule logic may be included as part of the function description, or recorded as a separate sub-function.

13.7.2.3. Other Custom Authorization Rules

There may be other business or system resources, such as specialized printers or forms, or message queues within the system context, which require authorization. Such authorization can be handled by identifying specific control functions to govern use of these resources, and providing controls to authorize the use of these

functions. However, it may be appropriate to capture the rules separately and distinctly, in relation to the resources themselves.

- *If authorization rules are required to control access to resources other than data and functions, the protected resource types must be represented as entity types on the system data model. The rules can then be represented as functions on the RULES hierarchy, with the prefix AUO.*
- *CRUD mappings should be identified appropriately for the entity types that represent the protected resources (n.b. these will not necessarily be included in the actual data base).*

13.7.3. Balance Control Rules

Balance control rules are generally used to validate that data has been processed completely and appropriately. This is usually done by taking summaries and counts before, during, and/or after the execution of a specific process or series of processes, and comparing them either individually or as combined by a specific algorithm. In general, the summaries are focused on arithmetic data, usually financial data or other quantities that have a direct bearing on costs or revenue. For purposes of validating completeness, summaries can also be applied as “hash totals” against numeric fields, such as account numbers, that have no arithmetic meaning.

Examples:

The sum of all closing balances on accounts must equal the sum of all opening balance, plus the sum of all receipts, minus the sum of all disbursements.

The hash total of account numbers on the transmission file from the cash processing system must match the control hash total transmitted with that file.

Balance control rules are generally established as part of the systems analysis and may not be complete until a substantial amount of the design has been done. They are also most commonly integrated with the definition of business processes or processing cycles, and may be well represented as small process steps in their own right. In such cases, they need to be part of the general specification of functions and processes, though perhaps isolated as sub-functions so that they are clearly visible.

There are occasionally situations where a balance control rule can be expressed as a static aggregate constraint on the database. This may happen where double entry book keeping is reflected in the data structure, or where data is being stored at different levels of aggregation.

- *In such a case, a balance control rule can be represented as a function on the RULES hierarchy, using the prefix BAL, and with reference ('R') CRUD to the data whose aggregate values are being validated by the rule.*

13.8. Information Integrity Rules

Information Integrity rules govern the production of aggregate and derived information products from the data base, including derived data calculations, and aggregations and transformations of data structures. They are intended to be used primarily in the specification of requirements for Decision Support systems. The descriptions that follow are provisional, particularly as these rules may be better specified and integrated with a tool suite that is more fully integrated with development of Decision Support solutions.

13.8.1. Derivation Rules

Derivation rules provide the algorithms used in deriving a value from the value or values of other attributes, constants, and/or dynamic parameters. The CRUD for the rule function should indicate reading of all attributes used in the algorithm. If the result is written to the data base, the CRUD should also include an insert (and/or update, as appropriate) to the resulting attribute.

- *Record derivation rules as functions, using the prefix DRV as the prefix in the function label.*

13.8.2. Data Matching Rules

Data matching rules provide the algorithms or logic used to determine if two objects relate to the same real world thing – frequently, the same person, or organization, or contract. They may simply indicate business identifiers to be used for precise matches, or they may involve specifications of various attributes to be considered, possibly with criteria weightings such as are used in probabilistic searching or matching software.

If a specific search or match function has been defined in the business or system function hierarchy, and no other functions have the same matching requirement, then a matching rule may be included as a dependent of the pertinent match or search function, but it should still be kept as a separate function specification. Any more widely used match criteria should appear in the RULES hierarchy.

- *Record change event rules as functions, using the prefix DMA as the prefix in the function label if the rule is recorded in the RULES hierarchy.*

13.8.3. Data Transformation Rules

Data transformation rules identify the ways in which specific data structures are transformed into other data structures, in support of specific conversions or interfaces. They may relate to the transformation of entities or even aggregates of entities, or of values from one domain into another, or of attributes in the context of a specific entity, aggregate, and/or domain transformation.

For purposes of conversions or interfaces, data transformation will likely have their own explicit position on the system function hierarchy. (They are unlikely to exist on the business function hierarchy, because they rarely involve the logical transformation of data content at all). The transformation rules for domains are likely to be more widely applicable, and so are the more likely candidates for inclusion in the RULES hierarchy.

- *Record data transformation rules as functions, using the prefix DTR as the prefix in the function label if the rule is recorded in the RULES hierarchy.*

13.9. Function Detail Rules

Some business functions are very rule intensive, with complex interactions among business rules that must all work together in order to enable the function to be performed or automated. In such cases, the rules are usually specific to the functions in which they are used. Examples from the Ministry of Education context include rules governing whether or not a student is a candidate to graduate, or whether and how an adult student is funded.

If the rules are applied in a clear sequence, they are easy to document as part of the function specification; but often the sequence in which these rules are applied is arbitrary, or varies with the data or context. In such a situation, or if the rules are somewhat volatile – some rules in the Education and Advanced Education sectors change virtually every year – it is advisable to document each rule separately.

For a relatively small number of rules per function, the rules should be recorded as sub-functions of the business functions to which they apply (i.e. NOT in the rules hierarchy). For situations where the number of rules for a specific function is in excess of about 8-10 (or even fewer if their interaction is complex), it is strongly recommended that some separate approach to rule modeling be chosen, and integrated with a rule implementation mechanism, as well as referred to from the function specification to which the rules apply.

14. Appendix A – Oracle Reserved Words

Table, view and element names cannot be ORACLE reserved names. ORACLE7 reserved names follow. This list may not be current, and should periodically be checked against whatever Oracle DBMS release is current at the time. It is current for release 9.0.2.3.

| | | |
|---------------|-----------------|-------------------------------|
| ABORT | AVAILABILITY | CHILD |
| ACCESS | BACKUP | CHOOSE |
| ACCESSED | BECOME | CHUNK |
| ACCOUNT | BEFORE | CLASS |
| ACTIVATE | BEGIN | CLEAR |
| ADD | BEHALF | CLOB |
| ADMIN | BETWEEN | CLONE |
| ADMINISTER | BFILE | CLOSE |
| ADMINISTRATOR | BINDING | CLOSE_CACHED_ OPEN_CURSORS |
| ADVISE | BITMAP | CLUSTER |
| AFTER | BITS | COALESCE |
| ALGORITHM | BLOB | COLLECT |
| ALIAS | BLOCK | COLUMN |
| ALL | BLOCKSIZE | COLUMNS |
| ALL_ROWS | BLOCK_RANGE | COLUMN_VALUE |
| ALLOCATE | BODY | COMMENT |
| ALLOW | BOUND | COMMIT |
| ALTER | BOTH | COMMITTED |
| ALWAYS | BROADCAST | COMPATIBILITY |
| ANALYZE | BUFFER_POOL | COMPILE |
| ANCILLARY | BUILD | COMPLETE |
| AND | BULK | COMPOSITE_LIMIT |
| ANY | BY | COMPRESS |
| APPLY | BYTE | COMPUTE |
| ARCHIVE | CACHE | CONFORMING |
| ARCHIVELOG | CACHE_INSTANCES | CONNECT |
| ARRAY | CALL | CONNECT_TIME |
| AS | CANCEL | CONSIDER |
| ASC | CASCADE | CONSISTENT |
| ASSOCIATE | CASE | CONSTRAINT |
| AT | CAST | CONSTRAINTS |
| ATTRIBUTE | CATEGORY | CONTAINER |
| ATTRIBUTES | CERTIFICATE | CONTENTS |
| AUDIT | CFILE | CONTEXT |
| AUTHENTICATED | CHAINED | CONTINUE |
| AUTHID | CHANGE | CONTROLFILE |
| AUTHORIZATION | CHAR | CONVERT |
| AUTO | CHAR_CS | CORRUPTION |
| AUTOALLOCATE | CHARACTER | COST |
| AUTOEXTEND | CHECK | CPU_PER_CALL |
| AUTOMATIC | CHECKPOINT | |

| | | |
|-----------------|---------------|---------------|
| CPU_PER_SESSION | DEREF | EXPLAIN |
| CREATE | DESC | EXPLOSION |
| CREATE_STORED_ | DETACHED | EXTEND |
| OUTLINES | DETERMINES | EXTENDS |
| CROSS | DICTIONARY | EXTENT |
| CUBE | DIMENSION | EXTENTS |
| CURRENT | DIRECTORY | EXTERNAL |
| CURRENT_DATE | DISABLE | EXTERNALLY |
| CURRENT_SCHEMA | DISASSOCIATE | EXTRACT |
| CURRENT_TIME | DISCONNECT | FAILED_LOGIN_ |
| CURRENT_ | DISK | ATTEMPTS |
| TIMESTAMP | DISKGROUP | FAILGROUP |
| CURRENT_USER | DISKS | FALSE |
| CURSOR | DISMOUNT | FAST |
| CURSOR_SPECIFIC | DISPATCHERS | FILE |
| _SEGMENT | DISTINCT | FILTER |
| CYCLE | DISTINGUISHED | FINAL |
| DANGLING | DISTRIBUTED | FINISH |
| DATA | DML | FIRST |
| DATABASE | DOUBLE | FIRST_ROWS |
| DATAFILE | DROP | FLAGGER |
| DATAFILES | DUMP | FLASHBACK |
| DATAOBJNO | DYNAMIC | FLOAT |
| DATE | EACH | FLOB |
| DATE_MODE | ELEMENT | FLUSH |
| DAY | ELSE | FOLLOWING |
| DBA | ENABLE | FOR |
| DBTIMEZONE | ENCRYPTED | FORCE |
| DDL | ENCRYPTION | FOREIGN |
| DEALLOCATE | END | FREELIST |
| DEBUG | ENFORCE | FREELISTS |
| DEC | ENTRY | FREEPOOLS |
| DECIMAL | ERROR_ON_ | FRESH |
| DECLARE | OVERLAP_TIME | FROM |
| DEFAULT | ESCAPE | FULL |
| DEFERRABLE | ESTIMATE | FUNCTION |
| DEFERRED | EVENTS | FUNCTIONS |
| DEFINED | EXCEPT | GENERATED |
| DEFINER | EXCEPTIONS | GLOBAL |
| DEGREE | EXCHANGE | GLOBALLY |
| DELAY | EXCLUDING | GLOBAL_NAME |
| DELETE | EXCLUSIVE | GLOBAL_TOPIC_ |
| DEMAND | EXECUTE | ENABLED |
| DENSE_RANK | EXEMPT | GRANT |
| ROWDEPENDENCIE | EXISTS | GROUP |
| S | EXPIRE | GROUPING |

| | | |
|--------------|------------------|----------------|
| GROUPS | INTERVAL | LOCATOR |
| GUARANTEED | INTO | LOCK |
| GUARD | INVALIDATE | LOCKED |
| HASH | IN_MEMORY_META | LOG |
| HASHKEYS | DATA | LOGFILE |
| HAVING | IS | LOGGING |
| HEADER | ISOLATION | LOGICAL |
| HEAP | ISOLATION_LEVEL | LOGICAL_READS_ |
| HIERARCHY | JAVA | PER_CALL |
| HOUR | JOIN | LOGICAL_READS_ |
| ID | KEEP | PER_SESSION |
| IDENTIFIED | KERBEROS | LOGOFF |
| IDENTIFIER | KEY | LOGON |
| IDGENERATORS | KEYFILE | LONG |
| IDLE_TIME | KEYS | MANAGE |
| IF | KEYSIZE | MANAGED |
| IMMEDIATE | REKEY | MANAGEMENT |
| IN | KILL | MANUAL |
| INCLUDING | << | MAPPING |
| INCREMENT | LAST | MASTER |
| INCREMENTAL | LATERAL | MATERIALIZED |
| INDEX | LAYER | MATCHED |
| INDEXED | LDAP_ | MAX |
| INDEXES | REGISTRATION | MAXARCHLOGS |
| INDEXTYPE | LDAP_REGISTRATIO | MAXDATAFILES |
| INDEXTYPES | N_ENABLED | MAXEXTENTS |
| INDICATOR | LDAP_REG_SYNC_ | MAXIMIZE |
| INITIAL | INTERVAL | MAXINSTANCES |
| INITIALIZED | LEADING | MAXLOGFILES |
| INITIALLY | LEFT | MAXLOGHISTORY |
| INITTRANS | LESS | MAXLOGMEMBERS |
| INNER | LEVEL | MAXSIZE |
| INSERT | LEVELS | MAXTRANS |
| INSTANCE | LIBRARY | MAXVALUE |
| INSTANCES | LIKE | METHOD |
| INSTANTIABLE | LIKE2 | MIN |
| INSTANTLY | LIKE4 | MEMBER |
| INSTEAD | LIKEC | MEMORY |
| INT | LIMIT | MERGE |
| INTEGER | LINK | MIGRATE |
| INTEGRITY | LIST | MINIMIZE |
| INTERMEDIATE | LOB | MINIMUM |
| INTERNAL_USE | LOCAL | MINEXTENTS |
| INTERNAL_ | LOCALTIME | MINUS |
| CONVERT | LOCALTIMESTAMP | MINUTE |
| INTERSECT | LOCATION | MINVALUE |

| | | |
|-----------------|---------------|-----------------|
| MIRROR | NLS_TERRITORY | OID |
| MLSLABEL | NO | OIDINDEX |
| MODE | NOARCHIVELOG | OLD |
| MODIFY | NOAUDIT | ON |
| MONITORING | NOCACHE | ONLINE |
| MONTH | NOCOMPRESS | ONLY |
| MOUNT | NOCYCLE | OPAQUE |
| MOVE | NOROWDEPENDEN | OPCODE |
| MOVEMENT | CIES | OPEN |
| MTS_DISPATCHERS | NODELAY | OPERATOR |
| MULTISET | NOFORCE | OPTIMAL |
| NAME | NOLOGGING | OPTIMIZER_GOAL |
| NAMED | NOMAPPING | OPTION |
| NATIONAL | NOMAXVALUE | OR |
| NATURAL | NOMINIMIZE | ORDER |
| NCHAR | NOMINVALUE | ORGANIZATION |
| NCHAR_CS | NOMONITORING | OUTER |
| NCLOB | NONE | OUTLINE |
| NEEDED | NOORDER | OVER |
| NESTED | NOOVERRIDE | OVERFLOW |
| NESTED_TABLE_ID | NOPARALLEL | OVERLAPS |
| NETWORK | NORELY | OWN |
| NEVER | NOREPAIR | PACKAGE |
| NEW | NORESETLOGS | PACKAGES |
| NEXT | NOREVERSE | PARALLEL |
| NLS_CALENDAR | NORMAL | PARAMETERS |
| NLS_ | NOSEGMENT | PARENT |
| CHARACTERSET | NOSTRICT | PARITY |
| NLS_COMP | NOSTRIPE | PARTIALLY |
| NLS_NCHAR_CONV | NOSORT | PARTITION |
| _EXCP | NOSWITCH | PARTITIONS |
| NLS_CURRENCY | NOT | PARTITION_HASH |
| NLS_DATE_FORMAT | NOTHING | PARTITION_LIST |
| NLS_DATE_ | NOVALIDATE | PARTITION_RANGE |
| LANGUAGE | NOWAIT | PASSWORD |
| NLS_ISO_ | NULL | PASSWORD_ |
| CURRENCY | NULLS | GRACE_TIME |
| NLS_LANG | NUMBER | PASSWORD_LIFE |
| NLS_LANGUAGE | NUMERIC | _TIME |
| NLS_LENGTH_ | NVARCHAR2 | PASSWORD_LOCK |
| SEMANTICS | OBJECT | _TIME |
| NLS_NUMERIC_ | OBJNO | PASSWORD_REUSE |
| CHARACTERS | OBJNO_REUSE | _MAX |
| NLS_SORT | OF | PASSWORD_REUSE |
| NLS_SPECIAL_ | OFF | _TIME |
| CHARS | OFFLINE | |

| | | |
|-----------------|--------------|-----------------|
| PASSWORD_ | REAL | ROWS |
| VERIFY_FUNCTION | REBALANCE | RULE |
| PCTFREE | REBUILD | SAMPLE |
| PCTINCREASE | RECORDS_PER_ | SAVEPOINT |
| PCTTHRESHOLD | BLOCK | SB4 |
| PCTUSED | RECOVER | SCAN |
| PCTVERSION | RECOVERABLE | SCAN_INSTANCES |
| PERCENT | RECOVERY | SCHEMA |
| PERFORMANCE | RECYCLE | SCN |
| PERMANENT | REDUCED | SCOPE |
| PFILE | REF | SD_ALL |
| PHYSICAL | REFERENCES | SD_INHIBIT |
| PLAN | REFERENCING | SD_SHOW |
| PLSQL_DEBUG | REFRESH | SECOND |
| POLICY | REGISTER | SECURITY |
| POST_ | REJECT | SEGMENT |
| TRANSACTION | RELATIONAL | SEG_BLOCK |
| PREBUILT | RELY | SEG_FILE |
| PRECEDING | RENAME | SELECT |
| PRECISION | REPAIR | SELECTIVITY |
| PREPARE | REPLACE | SEQUENCE |
| PRESERVE | RESET | SEQUENCED |
| PRIMARY | RESETLOGS | SERIALIZABLE |
| PRIOR | RESIZE | SERVERERROR |
| PRIVATE | RESOLVE | SESSION |
| PRIVATE_SGA | RESOLVER | SESSION_CACHED |
| PRIVILEGE | RESOURCE | _CURSORS |
| PRIVILEGES | RESTRICT | SESSIONS_PER_ |
| PROCEDURE | RESTRICTED | USER |
| PROFILE | RESUMABLE | SESSIONTIMEZONE |
| PROTECTED | RESUME | SESSIONTZNAME |
| PROTECTION | RETENTION | SET |
| PUBLIC | RETURN | SETS |
| PURGE | RETURNING | SETTINGS |
| PX_GRANULE | REUSE | SHARE |
| QUERY | REVERSE | SHARED |
| QUEUE | REVOKE | SHARED_POOL |
| QUIESCE | REWRITE | SHRINK |
| QUOTA | RIGHT | SHUTDOWN |
| RANDOM | ROLE | SIBLINGS |
| RANGE | ROLES | SID |
| RAPIDLY | ROLLBACK | SINGLE |
| RAW | ROLLUP | SINGLETASK |
| RBA | ROW | SIMPLE |
| READ | ROWID | SIZE |
| READS | ROWNUM | SKIP |

| | | |
|------------------|-----------------|---------------|
| SKIP_UNUSABLE_ | SYSDBA | UNBOUND |
| INDEXES | SYSOPER | UNBOUNDED |
| SMALLINT | SYSTEM | UNDER |
| SNAPSHOT | SYSTIMESTAMP | UNDO |
| SOME | TABLE | UNDROP |
| SORT | TABLES | UNIFORM |
| SOURCE | TABLESPACE | UNION |
| SPACE | TABLESPACE_NO | UNIQUE |
| SPECIFICATION | TABNO | UNLIMITED |
| SPFILE | TEMPFILE | UNLOCK |
| SPLIT | TEMPLATE | UNPACKED |
| SQL_TRACE | TEMPORARY | UNPROTECTED |
| STANDBY | TEST | UNQUIESCE |
| START | THAN | UNRECOVERABLE |
| STARTUP | THE | UNTIL |
| STATEMENT_ID | THEN | UNUSABLE |
| STATISTICS | THREAD | UNUSED |
| STATIC | THROUGH | UPD_INDEXES |
| STOP | TIMESTAMP | UPD_JOININDEX |
| STORAGE | TIME | UPDATABLE |
| STORE | TIMEOUT | UPDATE |
| STRIPE | TIMEZONE_ABBR | UPGRADE |
| STRICT | TIMEZONE_HOUR | UROWID |
| STRUCTURE | TIMEZONE_MINUTE | USAGE |
| SUBPARTITION | TIMEZONE_REGION | USE |
| SUBPARTITIONS | TIME_ZONE | USE_PRIVATE_ |
| SUBPARTITION_REL | TO | OUTLINES |
| SUBSTITUTABLE | TOplevel | USE_STORED_ |
| SUCCESSFUL | TRACE | OUTLINES |
| SUMMARY | TRACING | USER |
| SUSPEND | TRAILING | USER_DEFINED |
| SUPPLEMENTAL | TRANSACTION | USING |
| SWITCH | TRANSITIONAL | VALIDATE |
| SWITCHOVER | TREAT | VALIDATION |
| SYS_OP_BITVEC | TRIGGER | VALUE |
| SYS_OP_COL_ | TRIGGERS | VALUES |
| PRESENT | TRUE | VARCHAR |
| SYS_OP_ENFORCE | TRUNCATE | VARCHAR2 |
| _NOT_NULL\$ | TX | VARRAY |
| SYS_OP_MINE_ | TYPE | VARYING |
| VALUE | TYPES | VERSION |
| SYS_OP_ | TZ_OFFSET | VIEW |
| NOEXPAND | UB2 | WAIT |
| SYS_OP_NTCIMG\$ | UBA | WHEN |
| SYNONYM | UID | WHENEVER |
| SYSDATE | UNARCHIVED | WHERE |

WITH
WITHIN
WITHOUT
WORK
WRITE

XMLATTRIBUTES
XMLCOLATTVAL
XMLELEMENT
XMLFOREST
XMLTYPE

XMLSCHEMA
XID
YEAR
ZONE

15. Appendix B – Attribute Classes and Domain Names

This Appendix has been adapted from a similar one in the BC Ministry of Health’s *Business Requirements Modeling Standards and Guidelines*.

15.1. Standard class words and Root Domain names

As explained in the body of this document, all attributes should have names that end with a class word, and domains are to be organized and structured in relation to a predefined hierarchy of domain names. The following table identifies the standards for applicable class words and related root domain names. It is based on the ISO standards for data naming.

Class word abbreviations may be used where it is necessary to abbreviate the class word because of name length, or because of conventions of use (e.g., “ID” is generally used instead of the full word “Identifier”.)

Where a root domain name is followed by an “*”, this indicates that the domain is to be used as is, and is not to be subdivided.

| Class Category | Class Word | Abbr. | Root Domain | Definition | Examples |
|-----------------------|-----------------|-------|--|---|---|
| Label – Nominal Scale | Name | NM | Name | Name of a person or object | Last name Product name |
| | Identifier | ID | Identifier; ID* (for surrogate identifiers only) | Unique identifier of an entity. Business identifiers uniquely label business objects and may use mnemonic codes, such in a part id or product id. Surrogate identifiers usually have no meaning or intelligence. They are unique integers used to create unique table keys. | Product id Customer ID Account ID |
| | Number | NUM | Identifier | Non-quantifying alphanumeric string uniquely identifying a person, place, or thing. It does not express measurement or sequence. Arithmetic operations are not performed on numbers. Use num only if it is common usage (e.g. social insurance number, personal health number). | Telephone number Social insurance num |
| | Code | CD | Code | Alphanumeric character string representing a fact or property of an entity in a standard abbreviated or codified form | Customer type code Product type code |
| | Flag | FLG | Flag* | Binary (yes/no) property or condition. An identifier that has a domain of only 2 states: Y/N. | Deleted flg Sold flg |
| | [Other TBD] | [TBD] | Characteristic | Where the values in the domain are actual English words that describe an alternative set of qualities or characteristics – such as the actual English names of the colours – it is appropriate to use the actual name of the characteristic involved, such as the word ‘Colour’, and abbreviate it only if necessary. | Colour, Subject, School type |
| Label – Rank Scale | Sequence Number | SEQ | Integer | Number used to express the place of an object within a sequential ordering scheme | Report line seq |

| Class Category | Class Word | Abbr. | Root Domain | Definition | Examples |
|----------------------------|----------------|-------|---|--|--|
| | Rank | RNK | Integer, if assigned as an ordinal number, giving a unique position (other than tie situations) to every object in a set; otherwise Code. | Value used to express the relative position, priority, or importance of an object within a ranking scheme or hierarchy. | Military rank Credit score rank |
| Chronology | Date | DT | Date* | A point in time expressed as fully qualified date. | Birth date |
| | Time | TM | Time* | A unique point within a single day expressed as fully qualified time or hour, minute, second, or fraction of, in any combination (HH:MM:SS, HHMMSS, HH, etc.). The time class word does not include quantity of time, such as the number of minutes or hours to perform a task. Time quantities are represented with quantity class words and time modifiers (labor time qty, process time qty) | Start time End time Order received time End hour time |
| | Date Timestamp | DTS | Date Timestamp* | A unique point in time representing the fully qualified data and time. Expressed as CCYYMMDDHH:MM:SS | Row insert dts Last update dts |
| | Month | MNTH | Month* | The name of the month. | CALENDAR MNTH |
| | Quarter | QTR | Quarter | The quarter of the year. | CALENDAR QTR |
| | Year | YR | Year | The year expressed as a number. | CALENDAR YR |
| | Week | WK | Week | The week of the year expressed as a number. | CALENDAR WK |
| | Period | PRD | Period | A cyclical measure of time that is used for business measurement and/or management, and is not equivalent to any of the foregoing. For instance, many organizations divide the year into 13 4-week periods. | TIME PRD |
| Measurement Absolute Scale | Amount | AMT | Money | Monetary quantity; may also be used where the Unit of Measure is variable, and can include Money or currency as well as non-monetary units. | Loan original amt Loan payoff amt Allocation amt |
| | Balance | BAL | Money | Monetary account balance amount | Prin cur bal Avg mtd bal |
| | Fee | FEE | Money | Monetary fee amount | Late waived mtd fee Late assessed mtd fee |
| | Count | CNT | Integer | Non-monetary numeric value arrived at by counting Counts of transaction and activity occurrences | Withdrawal mtd cnt Deposit mtd cnt |

| Class Category | Class Word | Abbr. | Root Domain | Definition | Examples |
|-------------------------|---------------------|-------|--|--|---|
| | Quantity | QTY | Integer or Fixed Point | Non-monetary numeric value not arrived at by counting | Order qty Payment frequency qty |
| | Measure | MSR | Fixed Point or Floating Point | A physical measurement or scalable count such as size, weight, length, width, duration, latitude, etc. | Age msr Length msr Volume msr |
| | Unit of Measure | UOM | Unit of Measure (a subdomain of Code) | Unit of Measure that a measure, quantity, count, or amount is expressed in. (e.g. meters, months, days, dollars) | Payment frequenc qty uom |
| | Score | SCR | Fixed Point (if usable in calculation) ; otherwise Code | Score created by a predictive model or purchased from an information bureau | Behavioral current score Credit original score |
| Measurement Ratio Scale | Average | AVG | Fixed Point or Floating Point (belongs to the same domain as the data being averaged, or a closely related domain) | Arithmetic mean | Purchases mtd avg |
| | Percent | PCT | Percent | Number representing a relative measurement or ratio between two objects based on 100 | Discount pct |
| Description | Description | DESC | Text | A word or phrase describing the object (person, place, event, thing, etc.) identified by a label (identifier, name, code, etc.). Example: "Premium Personal Interest Free Checking Account" is describes product code PPIFC. | Product desc |
| | Text | TXT | Text | Any freeform comment or notes. The purpose of such free formtext, unlike name or description, may or may not be pre-determined.. | Street address text |
| Unstructured | Binary Large Object | BLOB | BLOB | A binary object containing an unstructured element of variable medium. | Test element blob |
| | Sound | SND | Sound | A binary object containing a sound(s). | Product desc snd |
| | Image | IMG | Image | Binary representations of a picture, chart, graph, photograph, etc. | Employee img Product img |

This list is not exhaustive, particularly with reference to specialized and unstructured data types. For other types not listed here, such as video streams or geographic coordinates and polygons, please consult with the Senior Data Architect in IMG.

15.2. Class word assignment guidelines

15.2.1. Identifiers and other types of labels

- ‘Identifier’ (ID) is the preferred class word for representing unique nominal labels (e.g., Customer ID). Use ‘number’ (NUM) as an alternative only where “number” is commonly used in business parlance when referring to the identifier in question (e.g. social insurance number, telephone number).
- ‘Identifier’ (other than the surrogate key ID) and ‘name’ should be used only where the object named or identified can be properly regarded as a person, place, event or thing – even if it is an abstraction (e.g. an associative structure). Otherwise, use ‘code’ or an English word identifying a characteristic, or one of the ranked label types.

15.2.2. Binary (yes/no) conditions versus Codes / Characteristics

- Use flg (Flag) if a data element represents a binary (yes/no) condition, and is not logically extensible.
- If a data element could, by extension represent more than two conditions, use cd (Code) or an English word identifying a characteristic. For instance, an account might initially be represented as either open or closed, but could logically be represented as active, suspended, pending, closed, etc.

15.2.3. Ordering/ranking schemes

- Use seq (Sequence Number) to represent the place of an object in a sequential ordering scheme.
- Use rnk (Rank) to represent a non-numeric value that indicates a relative position in a hierarchy or ranking scheme (e.g. letter grades: A, B, C+, C, C-, D, F).

15.2.4. Numeric values

- Use amt (Amount) for monetary values.
- Use cnt (Count) to represent the number of occurrences of an activity / event, or the number of persons/objects
- Use qty (Quantity) to represent how much in non-monetary terms.
- Use msr (Measurement) to represent physical dimensions (e.g., how large, heavy, long, or old).

15.2.5. Computed/derived/summary values

- Use pct (Percent) and avg (Average).
- Use qty (Quantity) for counts (e.g., withdrawals mtd qty).
- For totals, use ‘total’ as a qualifier word, followed by the appropriate class word (e.g., revenue total amt).

15.2.6. Alphanumeric strings

- Use nm (Name) to represent a person’s name or what an object or event is called.
- Use desc (Description) for descriptive text or explanations of codes.
- Use txt (Text) for free-form textual information.
- Use num (Number) for non-quantifying numeric strings.

16. Appendix C – Example Relationship Names

These are some examples of relationship name pairs. This is not an exhaustive list. (This list has been copied from the BC Ministry of Health's *Business Requirements Modelling Standards and Guidelines*.)

| A category association for | Categorized by |
|-----------------------------------|--------------------------|
| A decrease in | Affected by |
| A decrement in | Decrement by |
| A further definition of | Further defined by |
| A participant in | Conducted on |
| A potential site of | Performed at |
| A price defined for | Priced by |
| A reference to | In |
| A term for | To measure |
| A user in | Defined by |
| Acts as | Is role of |
| Affirms insurance coverage for | Has coverage affirmed by |
| An amendment to | Amended by |
| An effect on | Affected by |
| An example of | Embodied in |
| An increase in | Affected by |
| An increment in | Incremented by |
| Associated with | An area of interest to |
| Associated with | An interest in |
| Attached to | Inclusive of |
| Attached to | Labeled with |
| Authorized by | The authorization for |
| Based on | The basis for |
| Based on | The originator of |
| Billed to | Billed for |
| Bought via | For |
| Called by | For |
| Changed in | Of |
| Characterized by | The characteristics for |
| Charged with | Charged to |
| Classified by | The classification for |
| Composed of | Part of |
| Composed of | Part of |

| A category association for | Categorized by |
|-----------------------------------|----------------------------|
| Contacted at | The mechanism to contact |
| Contacted via | In the context of |
| Copied from | The original of |
| Defined as | Used to define |
| Defined by | Used to define |
| Defined by | Used to define |
| Defined in terms of | Defined for |
| Describe by | About |
| Described by | The definition for |
| Described by | The description of |
| Described by | Used to describe |
| Described by | Of |
| Employs | Is employed by |
| Expressed in | The basis for |
| For | Subject to |
| Fulfilled by | Used to fulfill |
| Further defined by | A further definition of |
| Further described by | Used to describe |
| Further described by | A further description of |
| Governs | Participates in |
| Governs | Is governed by |
| Has | Is associated with |
| Has as components | Is a component of |
| Has as primary participant | Participates as primary in |
| Has assigned to it | Is assigned to |
| Has reporting | Allows Reporting |
| Has reporting | Allows reporting |
| Implemented in | To carry out |
| In the preparation of | Created by |
| Included in | Comprised of |
| Includes | Provides opinion on |
| Input to | Fed by |
| Involved in | Established from |
| Is a member of | Has as a member |
| Is assigned to | Has assigned to it |

| A category association for | Categorized by |
|--|----------------------------------|
| Is contact for | Has contact |
| Is contact for | Provides contact |
| Is organized into | Is a component of |
| Is role of | Takes on role of |
| Is the parent of (parental relationship) | Is the child of |
| Is the purchaser of | Has |
| Is the spouse of (spousal relationship) | Is the spouse of |
| Issues | Is issued by |
| Located at | The location of |
| Made by | The creator of |
| Made from | Composed of |
| Manufactured by | The producer of |
| Modified by | A modification of |
| Monitored by | To monitor |
| Offers | Is offered by |
| Offers | Is offered by |
| Owns | Is owned by |
| Part of | Composed of |
| Part of | Detailed by |
| Part of | Composed of |
| Partially defined by | Part of the definition for |
| Partially defined by | A partial definition of |
| Participates as the primary in | Participates as the secondary in |
| Participates in | Has as participant |
| Played by | Given |
| Player of | Played by |
| Prepared by | The preparer of |
| Provides validation for | Is validated by |
| Published as | Of |
| Rated by | The rating for |
| Referenced by | For |
| Requested by | The requestor of |
| Responsible for | The responsibility of |
| Returned from | Returned to |
| Set the priority for | Prioritized by |

A category association for**Categorized by**

| | |
|---------------------|-------------------------|
| Set the status for | Defined by |
| Shipped from | Shipped to |
| Shipped from | The source of |
| Shipped to | The destination of |
| Stored in | The storage for |
| Subject to | A condition for |
| Subject to | Conducted on |
| Supplied through | A valid supplier for |
| Takes on role of | Is a |
| Takes role of | Is role of |
| The author of | Created by |
| The basis for | Based on |
| The basis for | Specified under |
| The category for | Categorized by |
| The definition of | Defined by |
| The definition of | Of |
| The description for | Described by |
| The description for | Described by |
| The destination of | The delivery for |
| The execution of | Carried out in |
| The fulfiller of | Taken by |
| The initiator of | Initiated by |
| The initiator of | Initiated by |
| The location for | Located at |
| The location of | Located at |
| The object of | Of |
| The payee for | Paid for by |
| The placer of | Placed by |
| The planned use of | Invoked as |
| The presence of | A part in |
| The producer of | Manufactured by |
| The purpose for | Used for the purpose of |
| The purpose for | Used for the purpose of |
| The rating for | Rated by |
| The recipient in | Received by |

A category association for**Categorized by**

| | |
|-------------------|----------------------|
| The recipient of | To |
| The reviewer of | Reviewed by |
| The seller in | Sold by |
| The seller of | A role for |
| The shipper in | Shipped by |
| The source of | Received from |
| The source of | Taken from |
| The source of | Recorded for |
| The storage for | Stored in |
| The submitter of | Submitted by |
| The trigger of | Triggered by |
| The trigger of | In response to |
| The use of | Used as |
| The warehouse for | Warehoused at |
| To Make | Built or repaired by |
| Used by | A usage of |
| Used by | In need of |
| Used to define | Defined from |
| Used to define | Defined to |
| Used to define | Described by |
| Used to monitor | Monitored by |