Common Statistical Production Architecture
Logical Information Model (LIM)

(Version 0.9 December 2015)

I. Introduction 2
II. Using LIM 3
A. Required use of LIM in the specification of a CSPA Service 3
B. Other ways to use LIM 4
III. Physical Representation of LIM objects 4
IV. The Logical Information Model 5
A. Base Package 6
B. Process Package 8
C. Data & structural metadata Package 13
D. Concepts Package 17
V. Further Development and Governance of LIM 20
Annex 1: LIM Design Principles v0.1 22
Annex 2: LIM Information Object Definitions and Attributes 23

I. Introduction

1. The Generic Statistical Information Model (GSIM) provides a common language for describing the information objects relevant to the statistical production process. Having a common language increases the ability to compare information within and between statistical organizations. The Common Statistical Production Architecture (CSPA) is a reference architecture for the official statistics industry, providing the blueprint for designing and building statistical services in a way that facilitates sharing and easy integration into statistical production processes within or between statistical organisations

2. At a sprint session held in Ottawa in February 2015, participants concluded that the gap between the conceptual nature of the GSIM and the practical implementation focus of the CSPA was too wide. To bridge this gap, a new layer, a Logical Information Model (LIM), was needed. The aim of the LIM is to help developers of CSPA-compliant services by translating the conceptual GSIM information objects into physical specifications of the information that flows in and out of statistical services.

3. LIM refines the conceptual definitions from GSIM, and describes the information objects and logical relationships required to support a CSPA service, in a manner which is consistent with GSIM and independent of the terminology used in existing standards such as SDMX (Statistical Data and Metadata eXchange)[1] and DDI (Data Documentation Initiative)[2]. It supports consistent use of SDMX, DDI and other implementation standards in reusable CPSA services, whilst also making it easier for any organisations that do not use SDMX or DDI to implement CSPA-compliant services.

4. One of the requirements for a CSPA Service to be included in the CSPA Statistical Service Catalogue is that it is compliant with the architecture and it is specified using the LIM.

5. A pragmatic approach has been followed during the development of the LIM. Only those parts actually needed by CSPA Services during 2015 being developed during that year. Other parts will be worked on as and when needed. The LIM is designed in accordance with the LIM Design Principles (outlined in Annex1).The scope of the LIM, and the current state of development at the end of 2015 are illustrated in the following table. The remainder of this document describes the parts of LIM that have been developed so far.

LIM Packages	**Status at end 2015**	
Base	Complete	
Business	Not started	Unlikely to be needed in the near future

[1] https://sdmx.org/
[2] www.ddialliance.org/

Process	Complete	
Data and Structural Metadata	Complete	
Referential metadata	Not started	
Exchange	Not started	
Questionnaire	Not started	To be worked on 2016
Concept	Complete	
Variable	Not started	To be worked on 2016

II. Using LIM

A. Required use of LIM in the specification of a CSPA Service

6. The LIM is used in the specification of the service. At this stage of design, the service will already have an approved Statistical Service Definition and the service will have been defined in terms of GSIM and the business process flow.

7. For a CSPA service to be considered compliant with the architecture and approved for inclusion in the CSPA Statistical Service Catalogue, it is necessary for the Service designer to explain what the expected interface is. The CSPA Service Specification template requires the data and the process logic and methods to be modelled in LIM.

8. To do this, the Service designer must identify the relevant LIM objects for the service interface. There are some objects that are likely to always be needed (for example process design, data structures). For each service, some lower level details may be required to determine the particular structure of some of the objects identified from LIM (for example the detailed structure of an identified *Code List*). The service design should identify the specific behaviours of the interface (i.e. would you return a dataset, or a reference to a dataset.) Knowing the architectural patterns to be used and the communication platform may affect which LIM objects are used in the interface.

9. Service Designers can use the Process Package to describe the methodology to be implemented by the Service Builder. There are a number of objects within LIM that are used to describe processes. These objects range from those used to design the Business Process, including the methods and rules used to define the actions encompassed in the process, through to the objects needed to encapsulate the execution-time processing, identifying the inputs used and the outputs produced at each step.

10. For a CSPA Service, these objects are useful tools to enable the complete description of the Service, the actions the Service will undertake and the inputs and outputs needed by the Service to complete these actions. These are documented in the Service Specification.

11. The distinction between the design-time and execution-time process objects indicates a focal point for the Service Designers and Service Builders. While this split isn't mutually exclusive between the two roles, generally speaking a Service Designer could develop a service using the LIM

objects for describing a process, while the Service Builder would focus on the execution-time objects.

12. For example, in the Service Specification document used to describe the CSPA Service, the *Process Step* objects would be used to encapsulate the information about what the Service will do. For the Service Builder, the *Process Input*s and Outputs identify what information is to be passed across the service boundaries for the service to perform its function (i.e. the service interface).

13. The service team will need to validate their proposal with the CSPA Implementation Group. In the case where the LIM does not include sufficient objects or attributes, this group can develop the model further to meet the needs of services. Once approved the Service Builder will take this specification and implement it based on the standard(s) that will be used.

B. Other ways to use LIM

14. Service Builders can use LIM to describe any (machine actionable) orchestration that is encapsulated by the Service. This will help others to understand the data models and process logic in the service (so they can use it) and reuse existing implementations of LIM and their physical representations. They can record these LIM/Process diagrams as appropriate.

15. Assemblers need to know the LIM objects so they can understand whether they can implement the service. They also need to understand the interface, so that they can determine if the service will be able to be integrated.

III. Physical Representation of LIM objects

16. The primary interest for the designer and the builder of a CSPA service is likely to be the physical specification of the information that will flow into and out of the service. The definition of the LIM and the specification of physical representations based on this logical model provide the way to translate agreed GSIM information objects into consistent, standards aligned, physical inputs and outputs for CSPA services.

17. For the Service Specification, we need LIM to describe information objects and the precise logical relationships between them in a manner which is consistent with GSIM. The primary interest for the builder of a CSPA service is likely to be the physical specification of the information that will flow into and out of the service. Depending on what information is being represented in practice, DDI and SDMX are currently expected to provide the primary basis for the physical representation of statistical information (e.g. data and metadata) in CSPA.

18. Logical modelling for CSPA will align to the maximum practical extent with the logical models associated with the candidate standards. In cases where complete alignment with existing standards is not practical, the usual decision will be for the LIM to align with one or other of the choices on a "best fit" basis. The following principles guide this decision.

- If there is a clear and unambiguous influence from an object in the logical model of another standard (such as DDI or SDMX) then:
    - If that object has mandatory attributes, those should be mandatory in the LIM or the LIM must provide guidance to default that attribute once implemented.
    - If that object has a relationship with another object that significantly affects its behaviour or use, that same relationship must exist in the LIM.
    - Future known changes to that object should be accommodated for in the LIM.

- Try to maintain alignment with equivalent objects in the supporting physical standards, developers will need this alignment in place in order to implement. This will also maximize the ability for developers to reuse available toolkits.

- Only 'improve' objects from underlying physical standards once you have exhausted options to have those changes implemented in the physical standard; if the physical standard is changed, then reflect that change; if the physical standard isn't changed, explain the reason for the improvement.

- Consider emerging and alternative standards when trying to identify the need for new objects in the LIM – other standards may have already expended the effort required to address the use case.

- Consider using existing mapping work (for example between DDI and SDMX,) to help align these standards with the LIM.

- Make efforts to be aware of proposed and agreed future changes to key physical standards like DDI and SDMX.

IV. The Logical Information Model

19. The LIM contains 9 packages which are based on the five GSIM Groups. The table below shows the relationship between the LIM packages and GSIM groups.

LIM Packages	GSIM Group
Base	Base
Business	Business
Process	
Data and structural metadata	Structures
Referential metadata	
Exchange	Exchange
Questionnaire	
Concept	Concepts
Variable	

20. The following sections detail those parts of the LIM that were developed in 2015. The focus of the text is to describe the use of the objects in a CSPA service and the differences from GSIM.

A. Base Package

21. Two new information objects were introduced in the LIM. These are *Change Event* and *Agent In Role*.

Change Event

22. The *Change Event* was introduced in order to have a general way to manage changes in the states of information objects.

23. Definition: A *Change Event* captures that a change has occurred. It identifies the information objects that have been affected, and the new information objects that have been created due to the change.

24. The *Change Event* has a date attribute which is used to convey the event time and a changeType attribute. In this release the attribute remains 'String' allowing for flexible use; however as more use cases are implemented, the expectation is that this will evolve into controlled vocabulary in order to accurately map to a recognized object lifecycle.

Agent In Role

25. *Agent In Role* was introduced in order to have a more flexible and extensible way of adding *Roles* to *Agents* without having to change existing information objects. The introduction of this object also provides a way to relate the *Agent*, *Role* and *Administrative Details* objects.

26. Definition: An *Agent* acting in a specific *Role*.

27. An *Agent In Role* may apply to either type of *Agent* - an *Organization* or *Individual*. The object is intended to reflect a single *Agent* acting in a single *Role* and as such is a very unambiguous representation. A common example would be to identify which *Individuals* or departments within an *Organization* provide administrative data. An *Agent In Role* is not in itself an *Identifiable Arteface*.

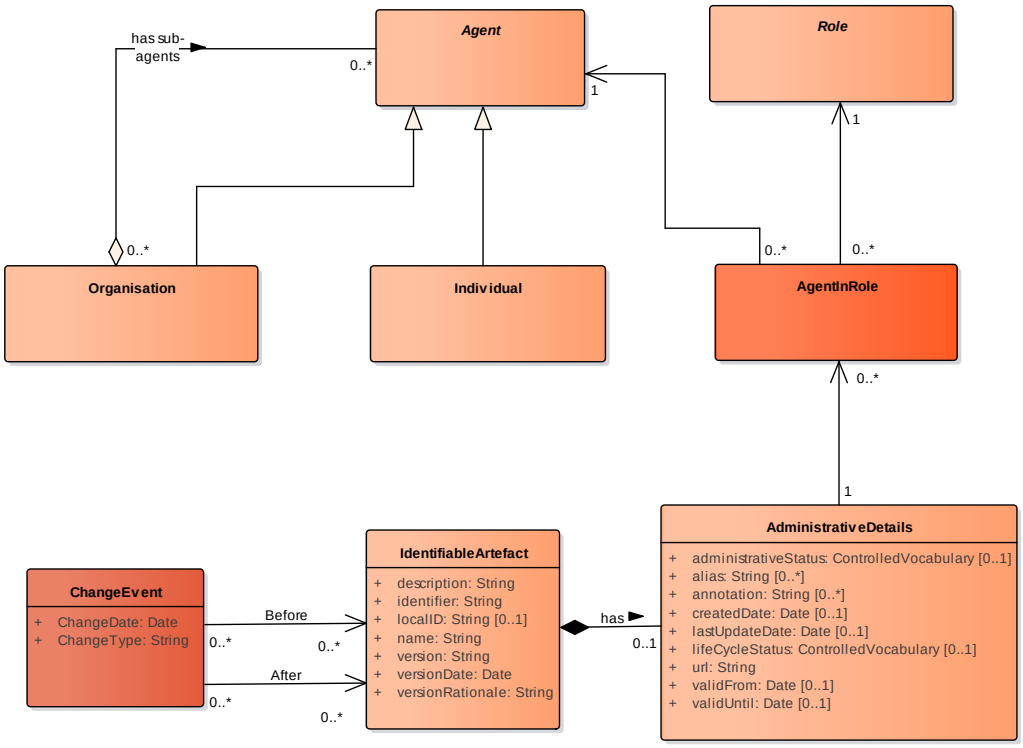

Figure 1: Base Group in LIM

28. Figure 1 shows the Base Group in LIM. Changes from GSIM v1.1 are described below:

Identifiable Artefact

29. An important decision was made that all attributes should be mandatory in *Identifiable Artefact*. The exception to this rule is Local ID. The consequence of this decision was a change in the cardinality of some attributes and the moving of some attributes to other Information objects.

After mapping to DDI, three new attributes were added (Local ID, Version Date and Version Rationale).

30. Two new usage relationships i.e. with the new information object *Change Event*.

31. The cardinality of the relationship to *Administrative Details* changed from 0 ... * to 0 ... 1. This both simplifies the model and provides focus for all non-mandatory attributes to be carried.

Agent

32. There has been one new usage association relationship introduced, from *Agent In Role* to *Agent*. Cardinality 0 ... * at *Agent In Role* and 1 at *Agent*. This replaces the relationship directly to Agent Role in GSIM v1.1

B. Process Package

33. There are a number of information objects defined in LIM to support the definition and execution of processes. These can be divided into two groups, those that are needed to design the process, and those that are used to capture the execution of a process.

Process Design

34. As part of the design of a service, the definition of the process that the service will perform is important. When the *Process Steps* are identified the next thing to do is to specify a *Process Design* for each step and identify how each *Process Step* will be performed. The result of the design, and the context and purpose for which the *Process Steps* will be used, are parts of the service specification that the service designer submits to the service builder.

35. *Process Designs* specify different types of inputs and outputs represented by the *Process Input Specification* and *Process Output Specification*. Examples of *Process Inputs* include data, metadata such as *Statistical Classifications*, *Rules* and *Parameters*. *Process Outputs* can be reports of various types (processing metrics), edited and/or new *Data Sets* and new or revised instances of metadata. The *Process Design* also consists of a *Process Method* which specifies the method to be used and is associated with a set of *Rules* to be applied.

36. The *Process Control Design* specifies the control logic, that is the sequencing and conditional flow logic within a *Process Step*/Service or between different *Process Steps*/Services. The flow describes what should happen next or what set of possible next steps can be performed after a specific *Process Step.*

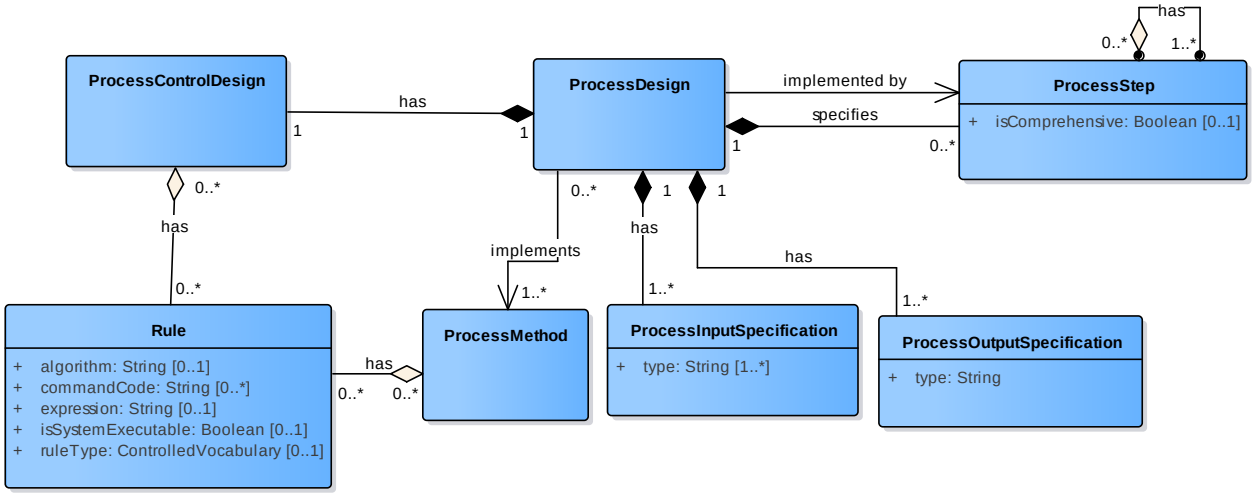

Figure 2: Process Design

Process Execution

37. There are four information objects that are central to the implementation and execution of a Service. These are *Process Step, Process Control, Process* Input and *Process Output*.

38. While these are the four most important objects to clearly define the requirements of a service, there are other objects which also provide useful information for developing a service. These may include *Process Step Instance* (to capture the 'real-time' execution of the *Process Step*), and *Process Design, Process Method* and *Rule* (to further define the requirements that are to be implemented by the *Process Step* and service).

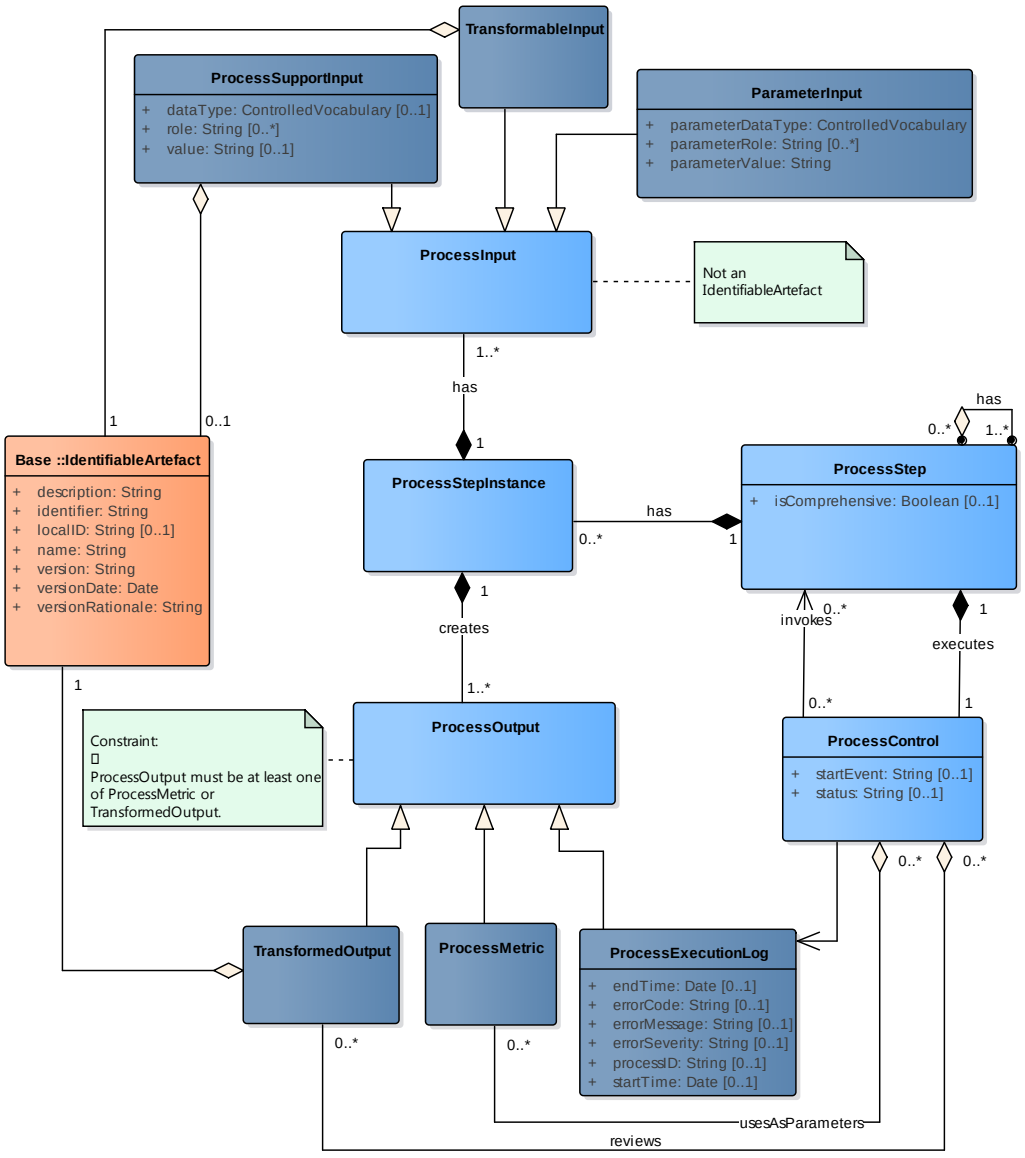

Figure 3: Process Execution

Moving from Design to Execution

39. Figure 4 shows the alignment between design-time and execution-time objects.

40. The most crucial aspect is that the execution-time *Process Step* encapsulates the whole *Process Design*, including the *Process Method* and *Rules*. The implementation of a *Process Step* will embed both the *Process Method* and *Rule*, where any configurable elements of the *Rule* will be supplied via the *Process Input* objects that are appropriate.

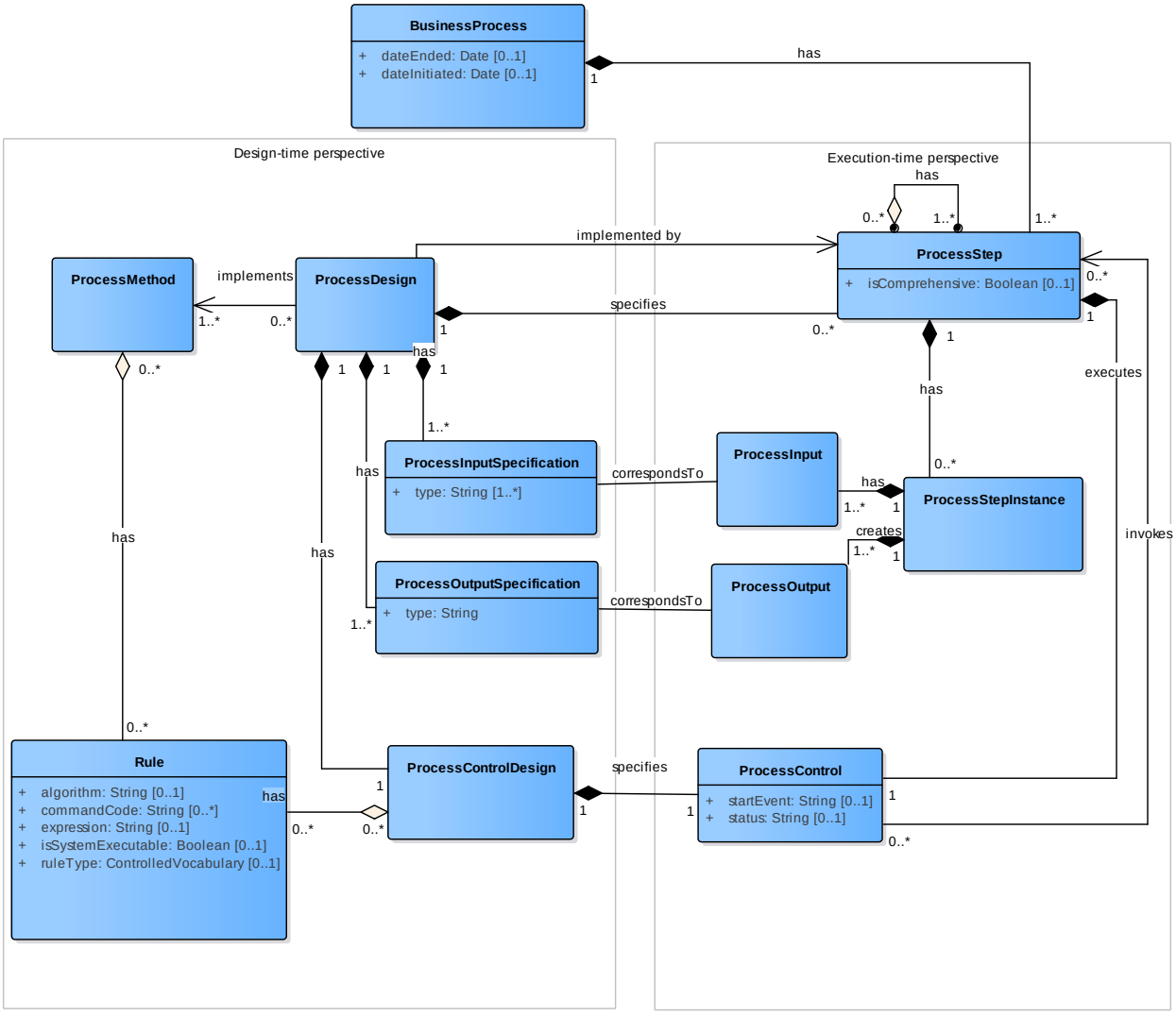

Figure 4: Alignment between design-time and execution-time

Process Step

41. GSIM Definition: A *Process Step* is a work package that performs a *Business Process*. A *Process Step* implements the *Process Step Design* specified in order to produce the outputs for which the *Process Step* was designed.

42. The *Process Step* outlines the required actions that will be executed by a Service. When designing and building the service, the information that was captured by the *Process Step* will specify what the Service is required to perform. This may also include information that was described in the *Process Design*, the *Process Method*, and the *Rule*.

43. Every time a *Process Step* is run, a *Process Step Instance* is created to capture the execution information, including the actual instances of the input information, and the resultant outputs.

Process Control

44. GSIM Definition: A set of decision points which determine the flow between the *Process Steps* used to perform a *Business Process*.

45. The *Process Control* defines the flow within the *Process Steps*, and also between different *Process Steps*. The *Process Control* can be implemented at different levels within a described process. It can describe the flow within a service, and also between different services.

46. Within a Service, a *Process Control* will be executed as part of a *Process Step*, and the *Process Control* will invoke the next *Process Step*.

47. *Process Input* and *Process Output* are used to identify the information and objects to be passed across the service boundary – that which is needed by the service to perform its function. There are three types of *Process Input* and three types of *Process Output*, which have been defined to specify the particular uses of information and objects in the service (input), and the different artefacts that are created as a result of the service (output). These objects were present in GSIM 1.0, but are not in the current version of GSIM (v1.1). They were reintroduced in LIM as they are very useful for Service Designers and Builders.

Process Input

48. GSIM Definition: Any instance of an information object which is supplied to a *Process Step Instance* at the time its execution is initiated.

49. *Process Input* might include information that is used as an input that will be transformed (e.g. a *Data Set*), information that is used to control specific parameters of the process (e.g. a *Rule*), and information that is used as reference to guide the process (e.g. a *Code List*).

50. There are three types of *Process Input* in LIM: *Transformable Input, Process Support Input* and *Parameter Input*.

Transformable Input

51. LIM Definition: A type of *Process Input* whose content goes into a *Process Step* and is changed in some way by the execution of that *Process Step*. Some or all of the content will be represented in the *Transformed Output*.

52. A *Transformable Input* is any object passed into a service that will be manipulated by the execution of the service. Typical *Transformable Inputs* are *Data Sets* and structural metadata (if changed by a process and needed to describe another output or as an object in their own right).

Process Support Input

53. LIM Definition: A form of *Process Input* that influences the work performed by the *Process Step*, and therefore influences its outcome.

54. A *Process Support Input* is a resource that is referenced or used to guide the service in completing its execution. This could include look up guides, background data, or classifiers used as part of the service.

55. Some examples of a *Process Support Input* could include:

- A *Code List* which will be used to check whether the *Codes* recorded in one dimension of a *Data Set* are valid, or
- An auxiliary *Data Set* which will influence imputation for, or editing of, a primary *Data Set* which has been submitted to the *Process Step* as the *Transformable Input*.

Parameter Input

56. LIM Definition: Inputs used to specify which configuration should be used for a specific *Process Step*, which has been designed to be configurable.

57. *Parameter Inputs* may be provided where *Rules* and/or *Business Service* interfaces associated with a particular *Process Step* have been designed to be configurable based on inputs passed in to the Service.

58. Some examples of a *Parameter Input* could include:

- Threshold values
- Ranges (such as upperLimit and lowerLimit)

Process Output

59. GSIM Definition: Any instance of an information object which is produced by a *Process Step* as a result of its execution.

60. *Process Output* might include information that has been transformed or created as part of the service execution, or an operational or methodological measure captured during the execution.

61. There are three types of *Process Output* in LIM: T*ransformed Output, Process Execution Log* and *Process Metric*.

Transformed Output
62. LIM Definition: A *Process Output* (a result) which provides the "reason for existence" for the *Process Step*.

63. A *Transformed Output* is the product of the actions that were executed by the Service. Typically, a *Transformed Output* could be considered the updated ("value added") version of one or more *Transformable Inputs* supplied to the Service.

64. For example, a *Process Step* with the *Business Function* of imputing missing values is likely to result, as its *Transformed Output*, in a *Data Set* where values that were missing previously have been imputed.

Process Execution Log
65. LIM Definition: The *Process Execution Log* captures the output of a *Process Step* which is not directly related to the *Transformed Output* it produced. It may include data that was recorded during the real time execution of the *Process Step*.

66. A *Process Execution Log* captures the information that is collected as part of a service for operational purposes. A typical example is to capture error conditions, or the time it took to complete the service.

67. Every time a *Process Step*/Service is run, a *Process Step* Instance is created to capture the "real-time" execution information of the *Process Step*, including the actual instances of inputted information, and the resultant outputs.

Process Metric
68. LIM Definition: A *Process Output* whose purpose is to measure and report quality or methodological aspects of the *Process Step* performed during execution.

69. A *Process Metric* records the quality or methodological measurements about the execution of a service. One purpose for a *Process Metric* may be to provide a quality measure related to the *Transformed Output*.

70. For example, statistical quality measures generated as part of a Service may include a measure of how many records were imputed, and a measure of how much difference, statistically, the imputed values make to the *Data Set* overall.

C. Data & structural metadata Package

71. Almost all services require *Data Sets* as a *Process Input* in order to do carry out meaningful processing. The LIM enables you to describe a *Data Set* and its internal structure in a standard way so that it can be processed by CSPA services.

72. A *Data Set* is a collection of *Data Points* that conform to a known structure described by a *Data Structure* and its components. A *Data Set* may be either a *Unit Data Set* or a *Dimensional Data Set*.

73. Unit data and dimensional data are perspectives on data. Although not typically the case, the same set of data could be described both ways. Sometimes what is considered dimensional data by one organization (for example, a national statistical office) might be considered unit data by another (for example, Eurostat where the unit is the member state). A particular collection of data need not be considered to be intrinsically one or the other. This matter of perspective is conceptual. In GSIM, the distinction is that a *Unit Data Set* contains data about *Units* and a *Dimensional Data Set* contains data about either *Units* or *Populations.*

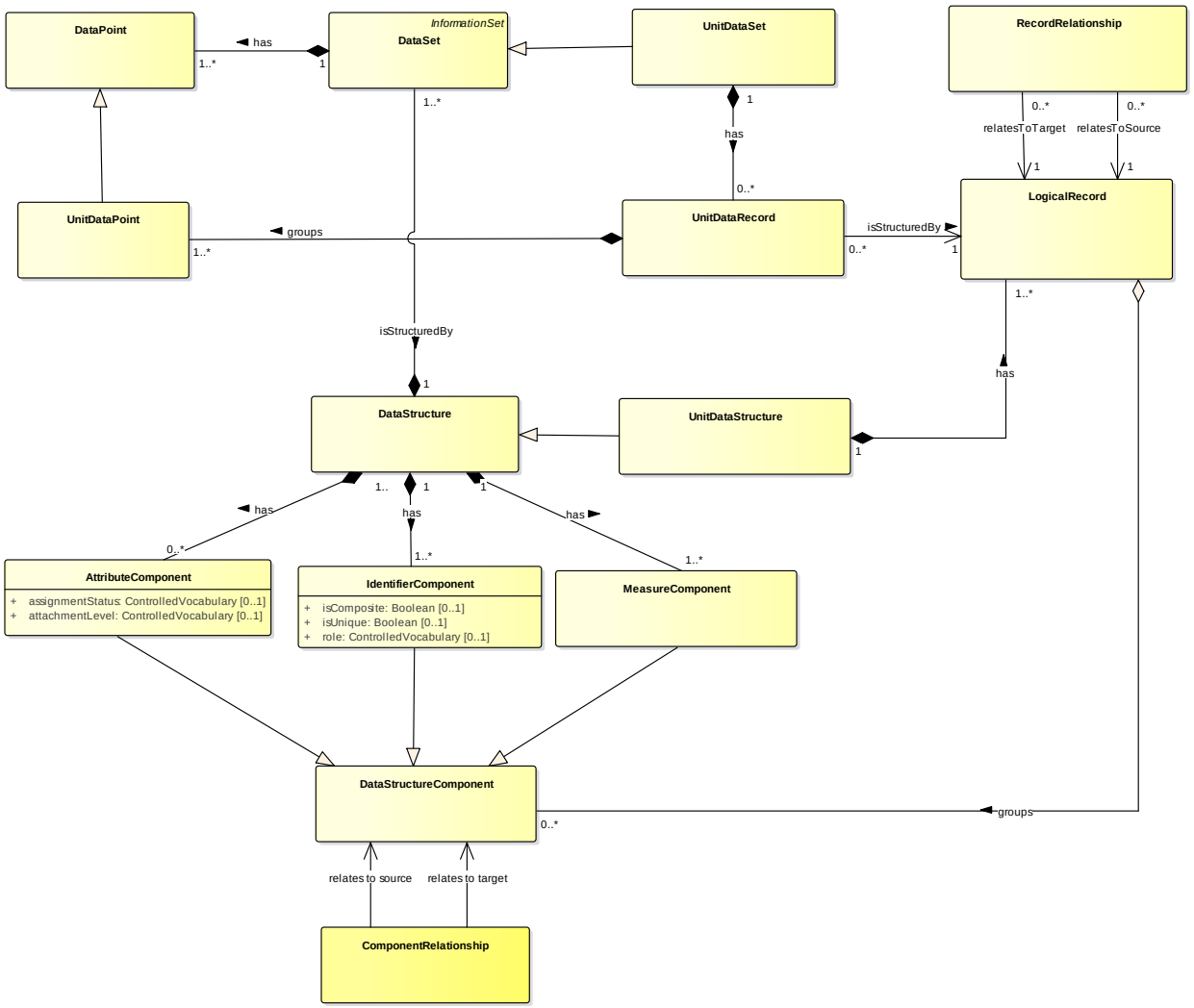

Figure 5: Unit Data Sets

74. A typical example of a *Unit Data Set* is census data. The model for *Unit Data Set* is shown in Figure 5.

75. A typical example of a *Dimensional Data Set* is an aggregated data cube, where the microdata have been summed. The model for *Dimensional Data Set* is shown in Figure 6.

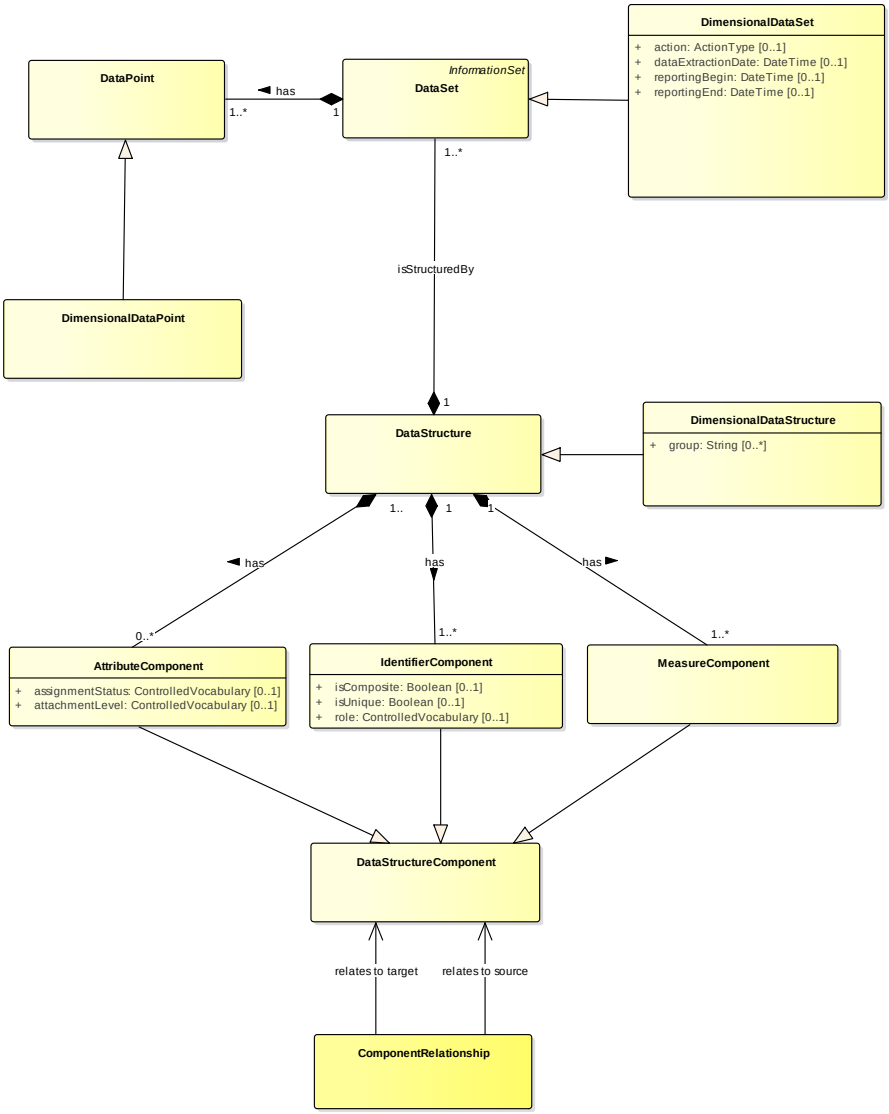

Figure 6: Dimensional Data Sets

76. One new object – *Component Relationship* -was added to LIM in the Data and Structural Metadata package.

77. This reflects that there could be a structure within the *Logical Record*, for example several fields can together represent a structured field (e.g. an address), or the record can be structured as in the case of an XML file conformant to a schema.

78. Changes from GSIM v1.1 are described below:

Data Structure

79. The composition relationship between *Data Structure Component* and *Data Structure* was removed for modelling purposes (coherence with the inheritance mechanism between the Components and *Data Structure Component* and its related cardinalities).

Attribute Component

80. A number of optional attributes were added to this object:

- The assignmentStatus attribute was added to indicate if an attribute defined in a *Data Structure* is optional or mandatory in a *Data Set*.
- An attachmentLevel attribute indicates the level to which an attribute is attached: it could be an observation, a dataset or series.

Identifier Component

81. A number of optional attributes were added to this object

- The isUnique attribute identifies if an *Identifier Component* uniquely identifies records in the *Data Sets* the corresponding *Data Structure* describes.
- The isComposite attribute specifies at the level of the *Data Structure Components* if the component alone is able to uniquely identify a record in the corresponding *Data Sets*.
- The role attribute is inspired from SDMX dimension roles, an initial controlled vocabulary proposed for this attributes contains the following 5 values: entity (e.g. personal identification number), identity (e.g. arbitrary number), count, time or geography.

82. Together the isUnique and isComposite attributes allow knowing if and how records can be uniquely identified in *Data Sets* corresponding to the *Data Structure*. This corresponds to the identification of unique keys and composite keys in ordinary databases. In LIM the fact that a key is composite can be derived by the number of *Identifier Components*, but it is believed it can be relevant to express at the level of an *Identifier Component* that it cannot identify a record by itself. The isUnique attribute adds information that cannot be derived from other elements.

83. The role attribute is used to characterize what is represented by an *Identifier Component*. For instance embedding in the LIM the identification of time and geography dimensions may allow for automated treatment of time/geography information based directly on the semantics of the model. This is believe to generate opportunities concerning automated treatment of LIM-compliant data independently of the specific content hosted by the model

Dimensional Data Set

84. A number of optional attributes were added to this object:

- The action attribute defines the action to be taken by the recipient system (replace, append, delete, information).
- The dataExtractionDate attribute describes the date and time that the data are extracted from a data source.
- The reportingBegin attribute describes the start date/time frame for the *Data Set*
- The reportingEnd attribute describes the end date/time frame for the *Data Set*

85. A Logical Information Model should stop at the logical level. When you translate the logical *Data Set* object from LIM into DDI or SDMX you are able to define more concrete physical aspects. This is conveyed in the Service Implementation document, not in the LIM.

D. Concepts Package

Statistical Classification

86. The Generic Statistical Information Model (GSIM): *Statistical Classifications* Model v1.1, defines the key concepts that are relevant to structuring *Statistical Classification* metadata, and provides the conceptual framework for the development of a *Statistical Classification* management system. It is aimed at classification experts.

87. LIM is aimed at service developers. In order to produce CSPA *Statistical Classification* services, seemingly redundant attributes have been removed i.e. replaced by relationships, and the number of attributes has been reduced to cover only those used by more than 60% of those organisations that have documented their use of GSIM *Statistical Classifications* in their GSIM Case studies (August 2015).

88. In LIM all information objects must use the attributes of the *Identifiable Artefact* and can use attributes from *Administrative Details* as needed. Changes in naming, cardinality and value type (e.g. text in GSIM is multilingual text in LIM) of a few attributes from GSIM *Statistical Classification* Model v1.1 have been made in LIM for greater consistency across all the information objects used and/or produced by CSPA services. For more details see comments per attribute below.

Statistical Classification

Name	Description	Cardinality	Value domain	Comment
description	The description provides details of the *Statistical Classification*, the background for its creation, the classification variable and objects/units classified, classification rules etc.	0...1	Multilingual text	Introduction and Description in GSIM are combined

validFrom	Date on which the *Statistical Classification* was released.	0....1	Date	Release Date in GSIM
validTo	Date on which the *Statistical Classification* was superseded by a successor version or otherwise ceased to be valid.	0....1	Date	Termination Date in GSIM

Level

Name	**Description**	**Cardinality**	**Value domain**	**Comment**
numberOf Items	The number of items (Categories) at the Level.	0...1	Integer	Text in GSIM

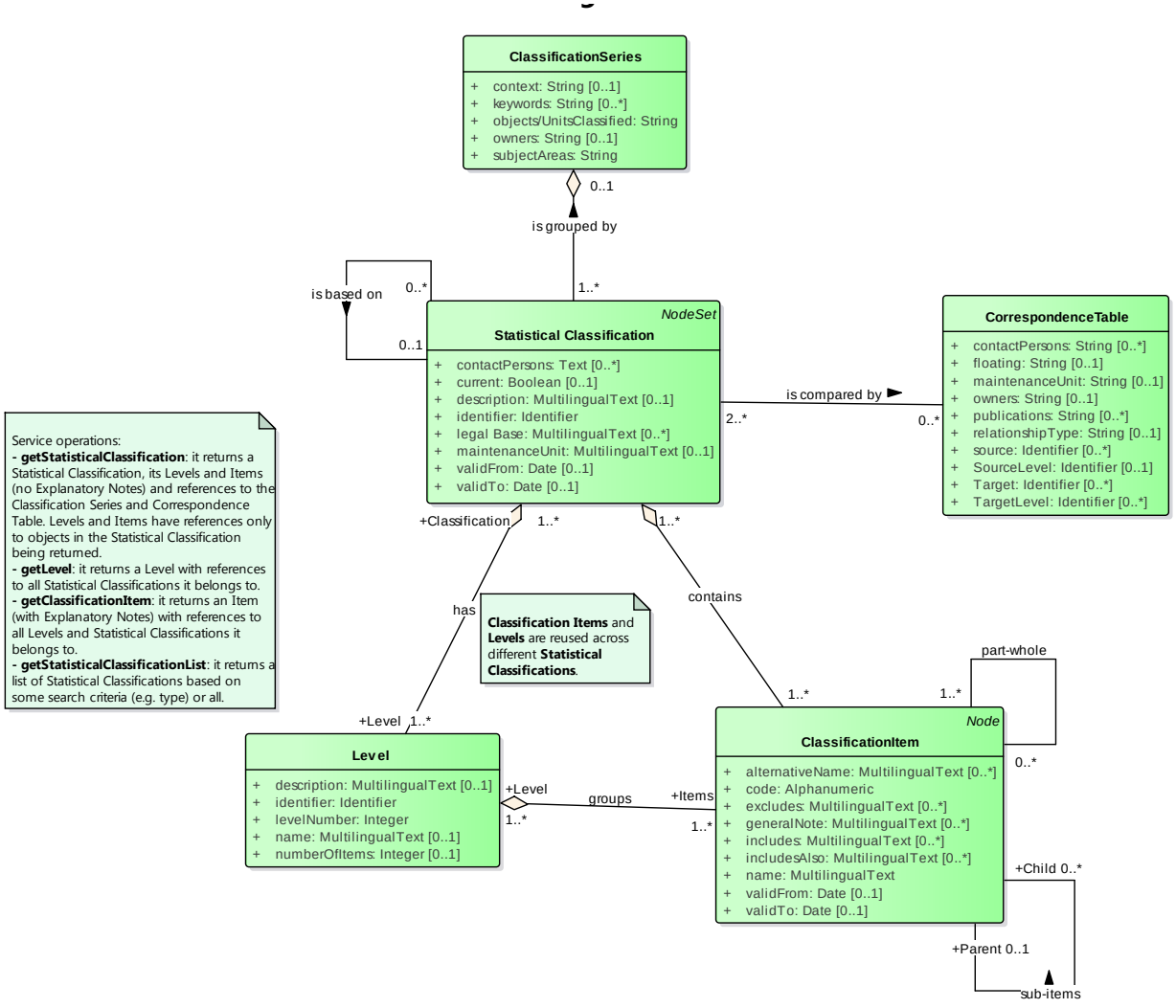

Figure 7: Statistical Classification

Code List

89. LIM for *Code Lists* started before LIM for *Statistical Classifications* in order to test out methods of working and the time needed to reach agreement for a run-time service. It may be necessary to expand the following minimal list of attributes and relationships when designing a *Code List* management system.

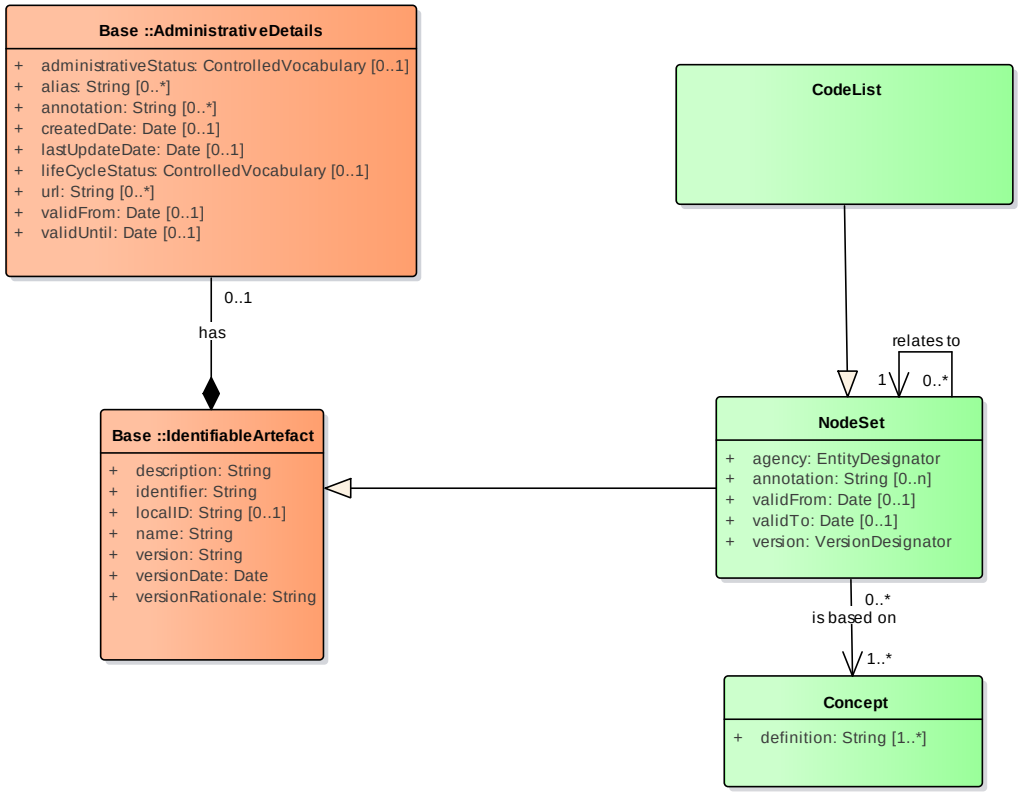

Figure 8: Code List

90. Note that in LIM a new relationship was introduced to the *Node Set* object allowing a *Node Set* to be 'basedOn' a *Concept*. This introduces the possibility of building a concept-based model for representing *Code Lists*, and is expected to further support the use of the LIM in future work regarding semantic technology.

Code Item

91. The *Code Item* has inherited a relationship to Concept via the *Node* object. As above (for *Code List*), this introduction allows a concept-based model for *Code Lists* and codes; it additionally supports any future use of concept mapping across *Data Sets* via the codes represented in them. For

the purposes of data integration, a semantic graph across concepts can now be theoretically navigated to inter-relate observations contained in different *Data Sets*.

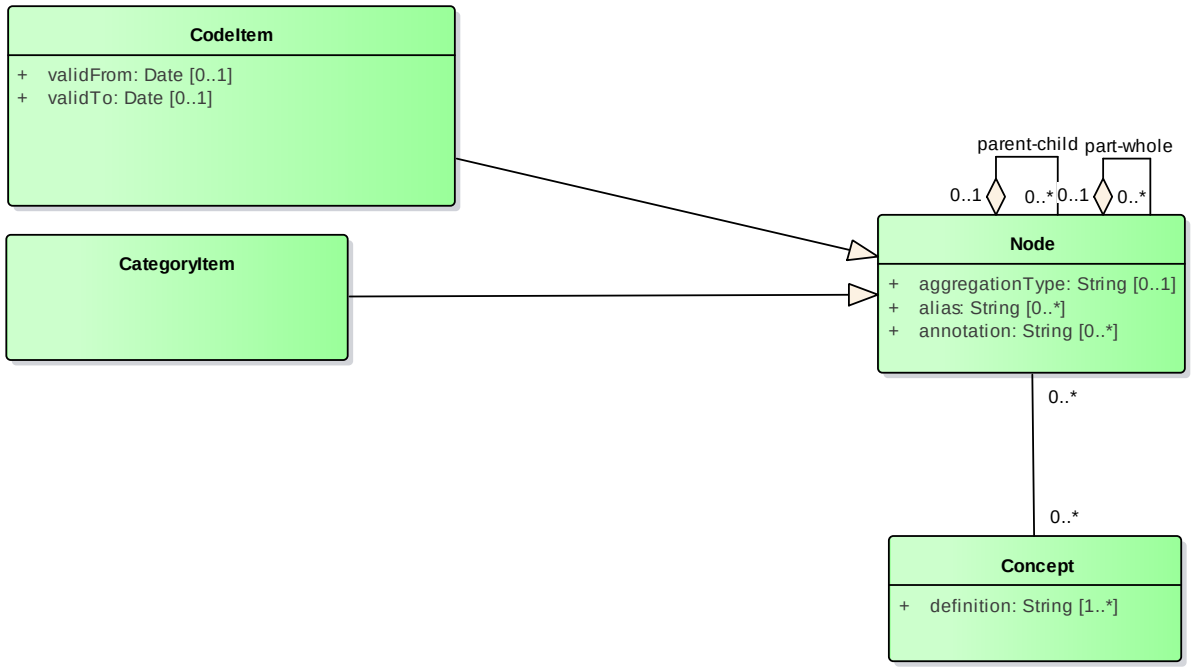

Figure 9: Code Item

V. Further Development and Governance of LIM

92. The further development of LIM and its physical representation will be undertaken incrementally based on business requirements. GSIM includes a wide variety of information objects and not all of them are required by services with the same urgency. In addition, the effort required to produce an all-encompassing logical model in a short time would be prohibitively expensive.

93. Not all GSIM information objects will make it to LIM. The LIM is only concerned with the service and as such will not be taking all GSIM information objects down to LIM level.

94. At any given point in time there will be CSPA services that require specific information objects for data exchange. Providing the logical and physical models for information objects in the design and implementation phases is critical. In the first instance the LIM is developed and provided to builders by the CSPA Implementation Group. This group is available to work with service designers and builders.

95. Any proposed changes to LIM will be confirmed through review by the CSPA community and mandated as standard within CSPA. This will add assurance that the recommended modelling provides a sound way forward. Additional attributes may be identified through consultation which

should be added to the model - but only if there is a common, practical, well defined use case for adding them. Following community review, the logical modelling will become "CSPA mandated".

96. Services which did not use the CSPA logical modelling (e.g. which provided an information interface based on DDI variant of the modelling where SDMX provided the basis for the modelling in CSPA) would not be CSPA compliant in regard to their inputs and outputs.

97. In this example, an agency which "preferred" DDI over SDMX might choose in their own environment to:

- translate from DDI to SDMX when providing inputs to the service, and
- translate from SDMX to DDI when accepting outputs from the service

98. Such translation would be the responsibility of the agency. While "adaptors" between common standards such as SDMX and DDI could be developed and shared in future to assist with this, CSPA services themselves would only accept input in a physical format which was fully consistent with the logical model.

Annex 1: LIM Design Principles v0.1

Principle 1: Change control.
Statement: LIM has change control and the following principles for designing LIM apply to every revision of LIM.

Principle 2: Backwards compatibility
Statement: LIM should provide maximum backwards compatibility.

Principle 3: Support for statistical services
Statement: LIM supports the design, development, production, maintenance and reuse of statistical services.

Principle 4: Common language
Statement: LIM provides a basis for a common understanding of information objects and the implementation of CSPA services

Principle 5: Appropriate communication
Statement: LIM must use appropriate communication for the target audience, in particular the language used in LIM must be precise and non-ambiguous.

Principle 6: Necessary level of detail
Statement: LIM contains a limited number of information objects for which the right balance between detail and abstraction should be used.

Principle 7: Robustness, adaptability and extensibility
Statement: LIM is robust, and can be easily adapted and extended to meet users' needs.

Principle 8: Optimal reuse
Statement: LIM makes optimal reuse of existing terms and definitions.

Principle 9: Platform independence
Statement: LIM does not refer to any specific IT setting or tool.

Annex 2: LIM Information Object Definitions and Attributes

A. Base

Administrative Details

Definition: A placeholder for extensions to the model based on an organisation's administrative needs.

Explanatory Text: The *Administrative Details* object is designed to act as a 'placeholder' to allow for future extensions to the existing model. It allows for further information to be added about the administrative details required to maintain the other objects outlined by GSIM.

Name	Description	Cardinality	Value Type
administrativeStatus	Indicator for access to an item: under review, open for use, or removed.	0..1	ControlledVocabulary
alias	The alias or alia associated with the information object.	0..*	String
annotation	A comment or instruction which provides additional explanations about the information object and how to use it.	0..*	String
createdDate	The date on which the information object was created	0..1	Date
lastUpdateDate	The date on which the information object was last changed.	0..1	Date
lifeCycleStatus	Indicator for the quality of an item: incomplete, valid, superseded, or retired	0..1	ControlledVocabulary
url	Allows location of the object. Distinct from the identification of the object as handled by the identifier attribute in *Identifiable Arteface*.	0..*	String

validFrom	The date on which the information object is effective or valid.	0..1	Date
validUntil	The date on which the information object is no longer effective or valid.	0..1	Date

Agent

Definition: An actor that performs a *Role* in relation to the statistical *Business Process*.

Explanatory Text: An *Agent* may be either an *Organization* or an *Individual*. An *Organization* may be an entire *Organization* or entities within a larger *Organization*, such as departments or divisions. An *Organization* may have sub *Agents*, which may be either other *Organizations* within the parent *Organization* or *Individuals* that belong to that *Organization*.

Agent In Role

Definition: Reflects an *Agent* acting in a specific *Role*.

Change Event

Definition: A *Change Event* captures that a change has occurred. It identifies the information objects that have been affected, and the new information objects that have been created due to the change.

Name	Description	Cardinality	Value Type
changeDate	The date on which the *Change Event* occurred	1..1	Date
changeType	The type of change that occurred	1..1	String
identifier	The unique identifier of the information object; assigned by the owner agency.	1..1	String

Identifiable Artefact

Definition: An abstract class that comprises the basic attributes and associations needed for identification, naming and other documentation.

Explanatory Text: An instance of any GSIM information object is an *Identifiable Artefact*, unless otherwise noted.

Name	Description	Cardinality	Value Type
description	The description of the information object	1..1	String
Identifier	The unique identifier of the information object; assigned by the owner agency.	1..1	String
localID	This is an identifier in a given local context that uniquely references an information object. For example, localId could be a variable name in a *Data Set*	0..1	String
Name	A term which designates a *Concept*, in this case an information object. The identifying name will be the preferred designation. There will be many terms to designate the same information object, such as synonyms and terms in other languages.	1..1	String
Version	The version designator of the information object assigned by the owner agency.	1..1	String
versionDate	The date on which the version was created.	1..1	Date
versionRationale	The reason for making this version of the information object.	1..1	String

Individual

Definition: A person who acts, or is designated to act towards a specific purpose.

Organisation

Definition: A unique framework of authority within which a person or persons act, or are designated to act, towards some purpose.

Role

Definition: The responsible function involved in the statistical *Business Process*.

Examples: Owner, Maintainer

B. Process

Business Process

Definition: The set of *Process Steps* to perform one of more *Business Functions* to deliver a *Statistical Program Cycle* or *Statistical Support Program.*

Explanatory Text: For example, a particular *Statistical Program Cycle* might include several data collection activities, the corresponding editing activities for each collection and the production and dissemination of final outputs. Each of these may be considered separate *Business Processes* for the *Statistical Program Cycle*.

Name	Description	Cardinality	Value Type
dateEnded	Last date of validity.	0..1	Date
dateInitiated	First date of validity.	0..1	Date

Business Service

Definition: A means of performing a *Business Function* (an ability that an organization possesses, typically expressed in general and high level terms and requiring a combination of organization, people, processes and technology to achieve).

Explanatory Text: A *Business Service* may provide one means of accessing a particular *Business Function*. The operation of a *Business Service* will perform one or more *Business Processes*.

The explicitly defined interface of a *Business Service* can be seen as representing a "service contract". If particular inputs are provided then the service will deliver particular outputs in compliance within specific parameters (for example, within a particular period of time).

Note: The interface of a *Business Service* is not necessarily IT based. For example, a typical postal service will have a number of service interfaces:

- Public letter box for posting letters
- Counter at post office for interacting with postal workers

Name	Description	Cardinality	Value Type
location	Specifies where the service can be accessed.	0..1	String
serviceInterface	Specifies how to communicate with the service.	0..*	String

Parameter Input

Definition: Inputs used to specify which configuration should be used for a specific *Process Step* which has been designed to be configurable.

Explanatory Text: *Parameter Inputs* may be provided where *Rules* and/or *Business Service* interfaces associated with a particular *Process Step* have been designed to be configurable based on inputs passed in to the *Process Step.*

Name	Description	Cardinality	Value Type
parameterDataType	The datatype of the parameter input	0..1	Controlled Vocabulary
parameterRole	Used to convey the *Role* of this parameter. For example: Weight, UpperThreshold, AgreementLevel	0..*	String
parameterValue	The content of the parameter	1..1	String

Process Control

Definition: A set of decision points which determine the flow between the *Process Steps* used to perform a *Business Process*.

Explanatory Text: The typical use of *Process Control* is to determine what happens next after a *Process Step* is executed. The possible paths, and the decision criteria, associated with a *Process Control* are specified as part of designing a production process, captured in a *Process Control Design*. There is typically a very close relationship between the design of a process and the design of a *Process Control*.

Name	Description	Cardinality	Value Type
startEvent	The event which triggered the control.	0..1	String
Status	Success or error, typically using a coded value.	0..1	String

Process Control Design

Definition: The specification of the decision points required during the execution of a *Business Process.*

Explanatory Text: The design of a *Process Control* typically takes place as part of the design of the process itself. This involves determining the conditional routing between the various sub-processes and services used by the executing process associated with the *Process Control* and specified by the *Process Control Design*.

It is possible to define a *Process Control* where the next step in the *Process Step* that will be executed is a fixed value rather than a "choice" between two or more possibilities. Where such a design would be appropriate, this feature allows, for example, initiation of a step in the *Process Step* representing the GSBPM Process Phase (5) to always lead to initiation of GSBPM sub-process Integrate Data (5.1) as the next step.

This allows a process designer to divide a *Business Process* into logical steps (for example, where each step performs a specific *Business Function* through re-use of a *Business Service*) even if these *Process Steps* will always follow each other in the same order. In all cases, the *Process Control Design* defines and the *Process Control* manages the flow between *Process Steps*, even where the flow is "trivial". *Process Design* is left to focus entirely on the design of the process itself, not sequencing between steps.

Process Design

Definition: The specification of how a *Process Step* will be performed. This includes specifying the types of *Process Input*s required and the type of *Process Output*s that will be produced.

Explanatory Text: A *Process Design* is the design time specification of a *Process Step* that is performed as part of a run-time *Business Service*. A *Process Step* can be as big or small as the designer of a particular *Business Service* chooses. From a design perspective, one *Process Step* can contain "sub-steps", each of which is conceptualized as a (smaller) *Process Step* in its own right. Each of those "sub-steps" may contain "sub-steps" within them and so on indefinitely. It is a decision for the process designer to what extent to subdivide steps. At some level it will be

appropriate to consider a *Process Step* to be a discrete task (unit of work) without warranting further subdivision. At that level the *Process Step* is designed to process particular *Process Input*s, according to a particular *Process Method*, to produce particular *Process Output*s. The flow between a *Process Step* and any sub steps is managed via *Process Control*.

Process Execution Log

Definition: The *Process Execution Log* captures the output of a *Process Step* which is not directly related to the *Transformed Output* it produced. It may include data that was recorded during the real time execution of the *Process Step*.

Name	Description	Cardinality	Value Type
endTime	The time the *Process Step* ended	0..1	Date
errorCode	The code for the error that occurred during the process execution.	0..1	String
errorMessage	The human readable message for the error that occurred during the process execution.	0..1	String
errorSeverity	The severity for the error that occurred during the process execution.	0..1	String
processID	Carries a reference to the instance, so that the log entry can be related to the process. Can be useful for both manual resolution or by the *Process Control*.	0..1	String
startTime	The time the *Process Step* started	0..1	Date

Process Input

Definition: Any instance of an information object which is supplied to a *Process Step* Instance at the time its execution is initiated.

Explanatory Text: *Process Input* might include information that is used as an input that will be transformed (e.g. a *Data Set*), information that is used to control specific parameters of the process (e.g. a *Rule*), and information that is used as reference to guide the process (e.g. a *Code List*).

Process Input Specification

Definition: A record of the types of inputs required for a *Process Design*.

Explanatory Text: The *Process Input* Specification enumerates the *Process Input*s required at the time a *Process Design* is executed. For example, if five different *Process Input*s are required, the *Process Input Specification* will describe each of the five inputs. For each required *Process Input* the *Process Input Specification* will record the type of information object (based on GSIM) which will be used as the *Process Input* (example types might be a *Dimensional Data Set* or a *Statistical Classification*).

The *Process Input* to be provided at the time of *Process Step* execution will then be a specific instance of the type of information object specified by the *Process Input* Specification. For example, if a *Process Input Specification* requires a *Dimensional Data Set* then the corresponding *Process Input* provided at the time of *Process Step* execution will be a particular *Dimensional Data Set*.

Name	Description	Cardinality	Value Type
type	This denotes the type of object which can be used as an input.	1..*	String

Process Method

Definition: A specification of the technique which will be used to perform the unit of work.

Explanatory Text: The technique specified by a *Process Method* is independent from any choice of technologies and/or other tools which will be used to apply that technique in a particular instance. The definition of the technique may, however, intrinsically require the application of specific *Rules* (for example, mathematical or logical formulas).

A *Process Method* describes a particular method for performing a *Process Step*.

Process Metric

Definition: A *Process Output* whose purpose is to measure and report some aspect of how the *Process Step* performed during execution.

Explanatory Text: A *Process Metric* is a sub-type of *Process Output* which records information about the execution of a *Process Step*. For example, how long it took to complete execution of the *Process Step* and what percentage of records in the *Transformable Input* was updated by the *Process Step* to produce the *Transformed Output.*

One purpose for a *Process Metric* may be to provide a quality measure related to the *Transformed Output*. For example, a *Process Step* with the *Business Function* of imputing missing values is likely to result, as its *Transformed Output*, in a *Data Set* where values that were missing previously have been imputed. Statistical quality measures, captured as *Process Metrics* for that *Process Step* may

include a measure of how many records were imputed, and a measure of how much difference, statistically, the imputed values make to the *Data Set* overall.

Another purpose for a *Process Metric* may be to measure an aspect of the *Process Step* which is not directly related to the *Transformed Output* it produced. For example, a *Process Metric* may record the time taken to complete the *Process Step* or other forms of resource utilization (for example, human and/or IT).

Often these two kinds of *Process Metrics* will be used in combination when seeking to, for example, monitor and tune a statistical *Business Process* so its statistical outputs achieve the highest level of quality possible based on the time, staff and/or IT resources that are available.

Process Output

Definition: Any instance of an information object which is produced by a *Process Step* as a result of its execution.

Process Output Specification

Definition: A record of the types of outputs required for a *Process Design*.

Explanatory Text: The *Process Output* Specification enumerates the *Process Output*s that are expected to be produced at the time a *Process Design* is executed. For example, if five different *Process Output*s expected, the *Process Output* Specification will describe each of the five outputs. For each expected *Process Output* the *Process Output* Specification will record the type of information object (based on GSIM) which will be used as the *Process Output* (Example types might be a *Dimensional Data Set* or a *Statistical Classification*).

The *Process Output* to be provided at the time of *Process Step* execution will then be a specific instance of the type of information object specified by the *Process Output* Specification. For example, if a *Process Output* Specification expects a *Dimensional Data Set* then the corresponding *Process Output* provided at the time of *Process Step* execution will be a particular *Dimensional Data Set*.

Name	Description	Cardinality	Value Type
Type	This denotes the type of object which can be used as an input.	1..*	String

Process Step

Definition: A *Process Step* is a work package that performs a Business Process. A *Process Step* implements the *Process Step* Design specified in order to produce the outputs for which the *Process Step* was designed.

Explanatory Text: Each *Process Step* is the use of a *Process Step* Design in a particular context (e.g. within a specific Business Process). At the time of execution a *Process Step* Instance specifies the actual instances of input objects (for example, specific *Data Sets*, specific *Variables*) to be supplied.

Name	Description	Cardinality	Value Type
isComprehensive	Used to indicate whether this *Process Step* has sub-*Process Steps*.	0..1	Boolean

Process Step Instance

Definition: An executed step in a *Business Process*. A *Process Step* Instance specifies the actual inputs to and outputs from for an occurrence of a *Process Step*.

Explanatory Text: Each *Process Step* is the use of a *Process Step Design* in a particular context (e.g. within a specific *Business Process*). At the time of execution a *Process Step Instance* specifies the actual instances of input objects (for example, specific *Data Sets*, specific *Variables*) to be supplied.

Each *Process Step Instance* may produce unique results even though the *Process Step* remains constant.

Even when the inputs remain the same, metrics such as the elapsed time to complete execution of *Process Step* may vary from execution to execution. For this reason, each *Process Step Instance* details of inputs and outputs for that instance of implementing the *Process Step*.

In this way it is possible to trace the flow of execution of a *Business Process* through all the *Process Steps* which were involved.

Process Support Input

Definition: A form of *Process Input* that influences the work performed by the *Process Step*, and therefore influences its outcome.

Explanatory Text: *Process Support Input* is a sub-type of *Process Input*. Typical *Process Support Inputs* include metadata resources such as *Statistical Classifications* or structural information used in the processing of data.

Examples of *Process Support Inputs* could include:

- A *Code List* which will be used to check whether the *Codes* recorded in one dimension of a *Data Set* are valid
- An auxiliary *Data Set* which will influence imputation for, or editing of, a primary *Data Set* which has been submitted to the *Process Step* as the *Transformable Input*.

In these examples, which *Code List* to use, or which auxiliary *Data Set* to use, may be specified via a *Parameter Input*. The details of the *Code List* or the auxiliary *Data Set* are *Process Support Inputs*.

Name	Description	Cardinality	Value Type
dataType	The datatype of the Process Support Input	0..1	Controlled Vocabulary
role	Used to convey the *Role* of this input.	0..*	String
value	The content of the Process Support Input	0..1	String

Rule

Definition: A specific mathematical or logical expression which can be evaluated to determine specific behaviour.

Explanatory Text: *Rules* are of several types: they may be derived from methods to determine the control flow of a process when it is being designed and executed; they may be used as the input parameters of processes (e.g., imputation rules, edit rules); and they may be used to drive the logical flow of a questionnaire. There are many forms of *Rules* and their purpose, character and expression can vary greatly.

Name	Description	Cardinality	Value Type
algorithm	The rule expressed as an algorithm.	0..1	String
commandCode	Structured information used by a system to process the instruction.	0..*	String
expression	The expression of the rule that is evaluated.	0..1	String
isSystemExecutable	Whether the rule is formatted to be executed by a system, or is only	0..1	Boolean

	documentary.		
ruleType	A type taken from a controlled vocabulary. For example: Input, Comparison, Imputation, Edit, Derivation, Recode	0..1	ControlledVocabulary

Transformable Input

Definition: A type of *Process Input* whose content goes into a *Process Step* and is changed in some way by the execution of that *Process Step*. Some or all of the content will be represented in the *Transformed Output.*

Explanatory Text: *Transformable Input* is a sub-type of *Process Input*. Producers of official statistics often conceptualize data (and sometimes metadata) flowing through the statistical *Business Process*, having statistical value added by each *Process Step* and being transformed along the way.

The concept of *Transformable Input* allows this notional flow of information through the production process to be traced, without confusing these inputs with other inputs - such as *Parameter Inputs* and *Process Support Inputs* that are controlling or influencing a particular *Process Step* but do not "flow through the business process" in the same sense. Typical *Transformable Inputs* are *Data Sets* and structural metadata (if changed by a process and needed to describe another output or as an object in their own right).

Transformed Output

Definition: A *Process Output* (a result) which provides the "reason for existence" for the *Process Step*.

Explanatory Text: A *Transformed Output* is a sub-type of *Process Output*. Typically a *Transformed Output* is either a *Process Input* to a subsequent *Process Step* or it represents the final product from a statistical business process.

In many cases a *Transformed Output* may be readily identified as an updated ("value added") version of one or more *Transformable Inputs* supplied to the *Process Step* execution.

C. Data & Structural Metadata

Attribute Component

Definition: The role given to a *Represented Variable* in the context of a *Data Structure*, which supplies information other than identification or measures.

Explanatory Text: For example the publication status of an observation (e.g. provisional, final, revised)

Name	Description	Cardinality	Value Type
assignmentStatus	When there is an attribute in a *Dimensional Data Structure*, this sets a status to indicate whether it is mandatory or optional to include it in that particular *Dimensional Data Set*	0..1	ControlledVocabulary
attachmentLevel	For a certain attachment, this describes at what level the attachment is at (*Data Set*, Observation, Series, Group)	0..1	ControlledVocabulary

Component Relationship

Definition: The role given to a *Represented Variable* in the context of a *Data Structure*, which supplies information other than identification or measures.

Explanatory Text: For example the publication status of an observation (e.g. provisional, final, revised)

Data Point

Definition: A placeholder (or cell) for the value of an *Instance Variable*

Explanatory Text: Field in a *Data Structure* which corresponds to a cell in a table. The *Data Point* is structural and distinct from the value (the *Datum*) that it holds.

Data Resource

Definition: An organized collection of stored information made of one or more *Data Sets*.

Explanatory Text: *Data Resources* are collections of data that are used by a statistical activity to produce information. *Data Resource* is a specialization of an *Information Resource*.

Data Set

Definition: An organized collection of data.

Explanatory Text: Examples of *Data Sets* could be observation registers, time series, longitudinal data, survey data, rectangular data sets, event-history data, tables, data tables, cubes, registers, hypercubes, and matrixes. A broader term for *Data Set* could be data. A narrower term for *Data Set* could be data element, data record, cell, field.

Data Structure

Definition: Defines the structure of an organized collection of data (*Data Set*).

Explanatory Text: The structure is described using *Data Structure Components* that can be either *Attribute Components*, *Identifier Components* or *Measure Components*. Examples for unit data include social security number, country of residence, age, citizenship, country of birth, where the social security number and the country of residence are both identifying components and the others are measured variables obtained directly or indirectly from the person (Unit).

Data Structure Component

Definition: The role of the *Represented Variable* in the context of a *Data Structure*.

Explanatory Text: A *Data Structure Component* can be an *Attribute Component*, *Measure Component* or an *Identifier Component*.

Example of *Attribute Component*: The publication status of an observation such as provisional, revised.

Example of *Measure Component*: age and height of a person in a *Unit Data Set* or number of citizens and number of households in a country in a *Data Set* for multiple countries (*Dimensional Data Set*).

Example of *Identifier Component*: The personal identification number of a Swedish citizen for unit data or the name of a country in the European Union for dimensional data.

Dimensional Data Point

Definition: A placeholder (or cell) for the value of an Instance Variable with respect to either a Unit or Population.

Explanatory Text: A *Dimensional Data Point* is uniquely identified by the combination of exactly one value for each of the dimensions (*Identifier Component*) and one measure (*Measure Component*). There may be multiple values for the same *Dimensional Data Point* that is for the same combination of dimension values and the same measure. The different values represent different versions of the data in the *Data Point*. Values are only distinguished on the basis of quality,

date/time of measurement or calculation, status, etc. This is handled through the mechanisms provided by the Datum information object.

Dimensional Data Set

Definition: A collection of dimensional data that conforms to a known structure.

Name	Description	Cardinality	Value Type
ActionType	Defines the action to be taken by the recipient system (replace, append, delete, information).	0..1	action
dataExtractionDat	A specific time period that identifies the date and time that the data are extracted from a data source.	0..1	DateTime
reportingBegin	A specific time period in a known system of time periods that identifies the start period of a report.	0..1	DateTime
reportingEnd	A specific time period in a known system of time periods that identifies the end period of a report.	0..1	DateTime

Dimensional Data Structure

Definition: Describes the structure of a *Dimensional Data Set*.

Explanatory Text: For example (city, average income, total population) where the city is the *Identifier Component* and the others are measured variables.

Name	Description	Cardinality	Value Type
group	A composite association to one or more component lists.	0..*	String

Identifier Component

Definition: The role given to a *Represented Variable* in the context of a *Data Structure* to identify the unit in an organized collection of data.

Explanatory Text: An *Identifier Component* is a sub-type of *Data Structure Component*. The personal identification number of a Swedish citizen for unit data or the name of a country in the European Union for dimensional data.

Name	Description	Cardinality	Value Type
isComposite	Indicates is the key is composite	0..1	Boolean
isUnique	Indicates if the key is unique	0..1	Boolean
role	specifies the type of id represented (entity, indicator, count, time, geography)	0..1	ControlledVocabulary

Logical Record

Definition: Describes a type of *Unit Data Record* for one *Unit Type* within a *Unit Data Set*.

Explanatory Text: Examples: household, person or dwelling record.

Measure Component

Definition: The role given to a *Represented Variable* in the context of a *Data Structure* to hold the observed/derived values for a particular *Unit* in an organized collection of data.

Explanatory Text: A *Measure Component* is a sub-type of *Data Structure Component*. For example age and height of a person in a *Unit Data Set* or number of citizens and number of households in a country in a *Data Set* for multiple countries (*Dimensional Data Set*).

Record Relationship

Definition: Describes relationships between *Logical Records* within a *Unit Data Structure*. It must have both a source *Logical Record* and a target *Logical Record* in order to define the relationship.

Explanatory Text: Example: Relationship between person and household *Logical Records* within a *Unit Data Set*.

Unit Data Point

Definition: A placeholder (or cell) for the value of an *Instance Variable* with respect to a *Unit*.

Explanatory Text: This placeholder may point to multiple values representing different versions of the data. Values are only distinguished on the basis of quality, date/time of measurement or calculation, status, etc. This is handled through the mechanisms provided by the *Datum* information object.

Unit Data Record

Definition: Contains the specific values (as a collection of *Unit Data Points*) related to a given *Unit* as defined in a *Logical Record*.

Explanatory Text: For example (1212123, 48, American, United Kingdom) specifies the age (48) in years on the 1st of January 2012 in years, the current citizenship (American), and the country of birth (United Kingdom) for a person with social security number 1212123.

Unit Data Set

Definition: A collection of data that conforms to a known structure and describes aspects of one or more *Units*.

Explanatory Text: Example: A synthetic unit record file is a collection of artificially constructed *Unit Data Records*, combined in a file to create a *Unit Data Set*.

Unit Data Structure

Definition: Describes the structure of a *Unit Data Set*.

Explanatory Text: For example (social security number, country of residence, age, citizenship, country of birth) where the social security number and the country of residence are the identifying components (*Identifier Component*) and the others are measured variables obtained directly or indirectly from the person (*Unit*) and are *Measure Components* of the *Logical Record*.

D. Concepts

Category

Definition: A *Concept* whose role is to extensionally define and measure a characteristic.

Explanatory Text: *Categories* for the *Concept* of sex include: Male, Female

Note: An extensional definition is a description of a *Concept* by enumerating all of its sub ordinate *Concepts* under one criterion or sub division.

For example - the Noble Gases (in the periodic table) is extensionally defined by the set of elements including Helium, Neon, Argon, Krypton, Xenon, Radon. (ISO 1087-1)

Category Item

Definition: An element of a *Category Set*.

Explanatory Text: A type of *Node* particular to a *Category Set* type of *Node Set*. A *Category Item* contains the meaning of a *Category* without any associated representation.

Category Set

Definition: A list of *Categories*

Explanatory Text: A *Category Set* is a type of *Node Set* which groups *Categories* through the use of *Category Items*. The *Categories* in a *Category Set* typically have no assigned *Designations* (*Codes*). For example: Male, Female

Classification Family

Definition: A *Classification Family* is a group of *Classification Series* related from a particular point of view. The *Classification Family* is related by being based on a common *Concept* (e.g. economic activity).

Explanatory Text: Different classification databases may use different types of *Classification Families* and have different names for the families, as no standard has been agreed upon.

Classification Index

Definition: A *Classification Index* is an ordered list (alphabetical, in code order etc.) of *Classification Index Entries*. A *Classification Index* can relate to one particular or to several *Statistical Classifications.*

Explanatory Text: A *Classification Index* shows the relationship between text found in statistical data sources (responses to survey questionnaires, administrative records) and one or more *Statistical Classifications*. A *Classification Index* may be used to assign the codes for *Classification Items* to observations in statistical collections.

A *Statistical Classification* is a subtype of *Node Set*. The relationship between *Statistical Classification* and *Classification Index* can also be extended to include the other *Node Set* types - *Code List* and *Category Set*.

Name	Description	Cardinality	Value Type
codingInstructions	Additional information which drives the coding process for all entries in a *Classification Index*.	0..*	String
corrections	Verbal summary description of corrections, which have occurred within the *Classification Index*. Corrections include changing the item code associated with an *Classification Index Entry*.	0..*	String

languages	A *Classification Index* can exist in several languages. Indicates the languages available. If a *Classification Index* exists in several languages, the number of entries in each language may be different, as the number of terms describing the same phenomenon can change from one language to another. However, the same phenomena should be described in each language.	0..*	String
maintenanceUnit	The unit or group of persons within the organisation responsible for the *Classification Index*, i.e. for adding, changing or deleting *Classification Index Entries*.	0..1	String
publications	A list of the publications in which the *Classification Index* has been published.	0..*	String
releaseDate	Date when the current version of the *Classification Index* was released.	0..1	Date

Classification Index Entry

Definition: A *Classification Index Entry* is a word or a short text (e.g. the name of a locality, an economic activity or an occupational title) describing a type of object/unit or object property to which a *Classification Item* applies, together with the code of the corresponding *Classification Item*.

Each *Classification Index Entry* typically refers to one item of the *Statistical Classification*. Although a *Classification Index Entry* may be associated with a *Classification Item* at any *Level* of a *Statistical Classification*, *Classification Index Entries* are normally associated with items at the lowest *Level*.

Explanatory Text: A *Classification Item* is a subtype of *Node*. The relationship between *Classification Item* and *Classification Index Entry* can also be extended to include the other *Node* types - *Code Item* and *Category Item*.

Name	Description	Cardinality	Value Type
codingInstructions	Additional information which drives the coding process. Required when	0..*	String

	coding is dependent upon one or many other factors.		
text	Text describing the type of object/unit or object property.	1..*	String
validFrom	Date from which the *Classification Index Entry* became valid. The date must be defined if the *Classification Index Entry* belongs to a floating *Classification Index*.	0..1	Date
validTo	Date at which the *Classification Index Entry* became invalid. The date must be defined if the *Classification Index Entry* belongs to a floating *Classification Index* and is no longer valid.	0..1	Date

Classification Item

Definition: A *Classification Item* represents a *Category* at a certain *Level* within a *Statistical Classification*. It defines the content and the borders of the *Category*. A *Unit* can be classified to one and only one item at each *Level* of a *Statistical Classification.*

Name	Description	Cardinality	Value Type
alternativeName	A *Classification Item* can be expressed in terms of one or several alternative names. Each alternative name is associated with a name type.	0..*	MultilingualText
code	A *Classification Item* is identified by an alphabetical, numerical or alphanumerical code, which is in line with the code structure of the classification *Level*. The code is unique within the *Statistical Classification* to which the item belongs.	1..1	Alphanumeric
excludes	A list of borderline cases, which do not belong to the described *Category*. Excluded cases may contain a reference to the *Classification Items* to which the	0..*	MultilingualText

	excluded cases belong.		
generalNote	Contains either additional information about the *Category*, or a general description of the *Category*, which is not structured according to the "includes", "includes also", "excludes" pattern.	0..*	MultilingualText
includes	Specifies the contents of the *Category*	0..*	MultilingualText
includesAlso	A list of borderline cases, which belong to the described *Category*.	0..*	MultilingualText
name	The name of the *Classification Item*	1..1	MultilingualText
validFrom	Date from which the item became valid. The date must be defined if the item belongs to a floating *Statistical Classification*	0..1	Date
validTo	Date at which the item became invalid. The date must be defined if the item belongs to a floating *Statistical Classification* and is no longer valid.	0..1	Date

Classification Series

Definition: A *Classification Series* is an ensemble of one or more *Statistical Classifications*, based on the same *Concept*, and related to each other as versions or updates. Typically, these *Statistical Classifications* have the same name (for example, ISIC or ISCO).

Name	Description	Cardinality	Value Type
context	*Classification Series* can be designed in a specific context.	0..1	String
keywords	A *Classification Series* can be associated with one or a number of keywords.	0..*	String
objects/UnitsClassified	A *Classification Series* is designed to classify a specific	1..1	String

	type of object/unit according to a specific attribute.		
owners	The statistical office or other authority, which created and maintains the *Statistical Classification*(s) related to the *Classification Series*. A *Classification Series* may have several owners.	0..1	String
subjectAreas	Areas of statistics in which the *Classification Series* is implemented.	1..1	String

Code

Definition: A *Designation* for a *Category*.

Explanatory Text: *Codes* are unique within their *Code List*. Example: M (Male) F (Female).

Code Item

Definition: An element of a *Code List*.

Explanatory Text: A type of *Node* particular to a *Code List* type of *Node Set*. A *Code Item* combines the meaning of the included *Category* with a *Code* representation.

Name	Description	Cardinality	Value Type
validFrom	Date from which the item became valid.	0..1	Date
validTo	Date at which the item became invalid.	0..1	Date

Code List

Definition: A list of *Categories* where each *Category* has a predefined *Code* assigned to it.

Explanatory Text: A kind of *Node Set* for which the *Category* contained in each *Node* has a *Code* assigned as a *Designation*. For example:
1 - Male
2 - Female

Code Value

Definition: An alpha-numeric string used to represent a *Code*.

Explanatory Text: A *Code Value* is a subtype of *Sign* - a way of denoting the value of a *Code*. This is a kind of *Sign* used for *Codes*.

Name	Description	Cardinality	Value Type
value	The value which is used to denote the *Code*	1..1	String

Concept

Definition: Unit of thought differentiated by characteristics.

Name	Description	Cardinality	Value Type
definition	Representation of a *Concept* by a descriptive statement which serves to differentiate it from related concepts.	1..*	String

Correspondence Table

Definition: A *Correspondence Table* expresses the relationship between two *Statistical Classification* s. These are typically: two versions from the same *Classification Series*; *Statistical Classifications* from different *Classification Series*; a variant and the version on which it is based; or, different versions of a variant. In the first and last examples, the *Correspondence Table* facilitates comparability over time. Correspondence relationships are shown in both directions.

Explanatory Text: A *Statistical Classification* is a subtype of *Node Set*. The relationship between *Statistical Classification* and *Correspondence Table* can also be extended to include the other *Node Set*s - *Code List* and *Category Set*.

Name	Description	Cardinality	Value Type
contactPersons	The person(s) who may be contacted for additional information about the *Correspondence Table*.	0..*	String
floating	If the source and/or target *Statistical Classifications* of a *Correspondence Table* are floating classifications, the date of the *Correspondence Table*	0..1	String

	must be noted. The *Correspondence Table* expresses the relationships between the two *Statistical Classifications* as they existed on the date specified in the table.		
maintenanceUnit	The unit or group of persons who are responsible for the *Correspondence Table*, i.e. for maintaining and updating it.	0..1	String
owners	The statistical office, other authority or section that created and maintains the *Correspondence Table*. A *Correspondence Table* may have several owners.	0..1	String
publications	A list of the publications in which the *Correspondence Table* has been published.	0..*	String
relationshipType	A correspondence can define a 1:1, 1:N, N:1 or M:N relationship between source and target items.	0..1	String
source	The *Statistical Classification* from which the correspondence is made.	0..*	Identifier
SourceLevel	The correspondence is normally restricted to a certain *Level* in the source *Statistical Classification*. In this case, target items are assigned only to source items on the given level. If no level is indicated, target items can be assigned to any level of the source *Statistical Classification*.	0..1	Identifier
Target	The *Statistical Classification* (s) to which the correspondence is directed. There may be multiple Target *Statistical Classifications* associated with the *Correspondence Table.*	0..*	Identifier
TargetLevel	The correspondence is normally	0..*	Identifier

	restricted to a certain *Level* in the target *Statistical Classification*. In this case, source items are assigned only to target items on the given *Level*. If no *Level* is indicated, source items can be assigned to any *Level* of the target *Statistical Classification*.		

Designation

Definition: The name given to an object for identification.

Explanatory Text: The association of a *Concept* with a *Sign* that denotes it.

Explanatory Notes

Definition: A *Classification Item* may be associated with explanatory notes, which further describe and clarify the contents of the *Category*. Explanatory notes consist of:

- **General note**: Contains either additional information about the *Category*, or a general description of the *Category*, which is not structured according to the "includes", "includes also", "excludes" pattern.
- **Includes:** Specifies the contents of the *Category*.
- **Includes also:** A list of borderline cases, which belong to the described *Category*.
- **Excludes:** A list of borderline cases, which do not belong to the described *Category*. Excluded cases may contain a reference to the *Classification Items* to which the excluded cases belong.

Name	Description	Cardinality	Value Type
excludes	A list of borderline cases, which do not belong to the described *Category*. Excluded cases may contain a reference to the *Classification Items* to which the excluded cases belong.	0..*	MultilingualText
generalNote	Contains either additional information about the *Category*, or a general description of the *Category*, which is not structured according to the "includes", "includes also", "excludes" pattern.	0..*	MultilingualText
includes	Specifies the contents of the Category	0..*	MultilingualText

includesAlso	A list of borderline cases, which belong to the described *Category*.	0..*	MultilingualText

Level

Definition: A *Statistical Classification* has a structure which is composed of one or several *Levels*. A *Level* often is associated with a *Concept*, which defines it. In a hierarchical classification the *Classification Items* of each *Level* but the highest are aggregated to the nearest higher *Level*. A linear classification has only one *Level*.

Explanatory Text: A *Statistical Classification* is a subtype of *Node Set*. The relationship between *Statistical Classification* and *Level* can also be extended to include the other *Node Set* types - *Code List* and *Category Set*.

Name	Description	Cardinality	Value Type
description	Text describing the content and particular purpose of the *Level*.	0..1	MultilingualText
identifier	The unique identifier of the information object.	1..1	Identifier
levelNumber	The number associated with the *Level*. *Levels* are numbered consecutively starting with level 1 at the highest (most aggregated) *Level*.	1..1	Integer
name	The name given to the *Level*.	0..1	MultilingualText
numberOfItems	The number of items (*Categories*) at the *Level*.	0..1	Integer

Map

Definition: A *Map* is an expression of the relation between a *Classification Item* in a source *Statistical Classification* and a corresponding *Classification Item* in the target *Statistical Classification*. The *Map* should specify whether the relationship between the two *Classification Items* is partial or complete. Depending on the relationship type of the *Correspondence Table*, there may be several *Maps* for a single source or target item.

Explanatory Text: A *Classification Item* is a subtype of *Node*. The relationship between *Classification Item* and *Map* can also be extended to include the other types of *Node* - *Code Item* and *Category Item*.

Name	Description	Cardinality	Value Type
validFrom	Date from which the *Map* became valid. The date must be defined if the *Map* belongs to a floating *Correspondence Table*.	0..1	Date
validTo	Date at which the *Map* became invalid. The date must be defined if the *Map* belongs to a floating *Correspondence Table* and is no longer valid.	0..1	Date

Node

Definition: A combination of a *Category* and related attributes.

Explanatory Text: A *Node* is created as a *Category, Code* or *Classification Item* for the purpose of defining the situation in which the *Category* is being used.

Name	Description	Cardinality	Value Type
aggregationType	To define the parent/child relationship between *Node*s, it tells us whether we are applying the part whole relationship, or the super/sub type relationships.	0..1	String
alias	A type of explanatory note that can be used to define alternative labels for the category contained within the *Node*.	0..*	String
annotation	A human-readable internal note intended for the developers/maintainers of LIM.	0..*	String

Node Set

Definition: A set of *Node*s.

Explanatory Text: *Node Set* is a kind of *Concept System*. Here are 2 examples:

- Sex Categories: Male, Female, Other
- Sex Codes: <m, male>, <f, female>, <o, other>

Name	Description	Cardinality	Value Type
agency	The organization or legal entity which owns and maintains the object.	1..1	EntityDesignator
annotation	A human-readable internal note intended for the developers/maintainers of LIM.	0..*	String
validFrom	The effective date on which the object is published.	0..1	Date
validTo	The effective date on which the object is withdrawn from publication.	0..1	Date
version	The version of the object assigned by the owning agency.	1..1	VersionDesignator

Sign

Definition: Something that suggests the presence or existence of a fact, condition, or quality.

Explanatory Text: It is a perceivable object. This object is used to denote a *Concept* as a *Designation*.

Name	Description	Cardinality	Value Type
Value	A human-readable value for the object.	1..1	String

Statistical Classification

Definition: A *Statistical Classification* is a set of *Categories* which may be assigned to one or more variables registered in statistical surveys or administrative files, and used in the production and dissemination of statistics. The *Categories* at each *Level* of the classification structure must be mutually exclusive and jointly exhaustive of all objects/units in the population of interest.

Explanatory Text: The *Categories* are defined with reference to one or more characteristics of a particular population of units of observation. A *Statistical Classification* may have a flat, linear structure or may be hierarchically structured, such that all *Categories* at lower *Levels* are sub-*Categories* of *Categories* at the next *Level* up. *Categories* in *Statistical Classifications* are represented in the information model as *Classification Items*.

Name	Description	Cardinality	Value Type
contactPersons	Person(s) who may be contacted for additional information about the *Statistical Classification*.	0..*	Text
Current	Indicates whether or not the *Statistical Classification* is currently valid.	0..1	Boolean
Description	Text describing the content and particular purpose of the *Statistical Classification* .	0..1	MultilingualText
Identifier	The unique identifier of the information object.	1..1	Identifier
legalBase	Indicates that the *Statistical Classification* is covered by a legal act or by some other formal agreement.	0..*	MultilingualText
maintenanceUnit	The unit or group of persons within the organisation who are responsible for the *Statistical Classification* (i.e., for maintaining, updating and changing it).	0..1	MultilingualText
validFrom	The effective date on which the object is published.	0..1	Date
validTo	The effective date on which the object is withdrawn from publication.	0..1	Date