DDI Alliance Scientific Board Annual Meeting
May 18, 2020, 13:00-15:00 UTC

Join Zoom Meeting:
https://us02web.zoom.us/j/86002275516?pwd=eVV2dmJzQkhXZ1Rja2FRVUZ2UGFGQT09
Meeting ID: 860 0227 5516
Password: 499786
Find your local number: https://us02web.zoom.us/u/kcySGSOm9

Agenda - Scientific Board Meeting			
Time	**Subject**	**Detail**	**Lead**
13:00-13:05	Welcome		Achim Wackerow, *Scientific Board Chair*
13:05-13:30	Scientific Board restructuring recommendations [slides]	-Scientific Board Review and Restructuring -Scientific Board Operational Guidelines -Proposed Bylaws changes -Discussion	Ingo Barkow, *Scientific Board Vice-Chair* *Achim, facilitator*
13:30-13:55	DDI-Cross Domain Integration (DDI-CDI) [slides]	-Features and Status report -Public review -Future direction -Discussion	Arofan Gregory, *MRT Group moderator* *Achim, facilitator*
13:55-14:20	Technical Committee [slides]	-TC Report for 2019-2020 -DDI Lifecycle - 3.3 publication -Expansion of DDI Lifecycle - Technical input needed -DDI Codebook - current and future revisions -Roadmap -Call for new TC members -Discussion	Wendy Thomas, *Technical Committee Chair* *Ingo, facilitator*
14:20-15:00	Scientific Board direction and goals for the year [slides]	-What are the goals? -What is the work plan? -Future of the Moving Forward project -Discussion	Achim Wackerow, *Scientific Board Chair* *Ingo, facilitator*

Other reports:

- DDI Controlled Vocabularies working group, 2019-2020 activity report

DDI Alliance Scientific Board Annual Meeting

May 18, 2020

13:00-15:00 UTC (online)

Agenda

Time	Subject	Detail	Lead
13:00-13:05	Welcome		Achim Wackerow, *Scientific Board Chair*
13:05-13:30	Scientific Board restructuring	• Recommendations to improve the Scientific Board structure • Discussion	Ingo Barkow, *Scientific Board Vice-Chair*
13:30-13:55	DDI-Cross Domain Integration (DDI-CDI)	• Public review • Future direction on DDI-CDI • Discussion	Arofan Gregory, *MRT Group moderator*
13:55-14:20	Technical Committee	• DDI Lifecycle - 3.3 publication • Call for new TC members • Expansion of DDI Lifecycle Technical input needed • DDI Codebook - current and future revisions • Discussion	Wendy Thomas, *Technical Committee Chair*
14:20-15:00	Scientific Board direction and goals for the year	• What are the goals? • What is the work plan? • Future of the Moving Forward project • Discussion	Achim Wackerow, *Scientific Board Chair*

Change of Scientific Board Structure

- Motivation: experience shows that the current set up - a Scientific Board of approx. 40 member representatives has limited impact regarding the role of the Scientific Board in the bylaws
  - Contribute to the substantive content of DDI standards and semantic products and approve major version revisions.
  - Evaluate technical proposals through the Alliance standards review process.
  - Undertake research and testing concerning proposals for DDI standards and semantic products.
  - Develop and promulgate best practices for use of DDI standards and semantic products.
  - Assess progress and barriers to progress.
  - Suggest future directions and activities for the Alliance.
- In 2019 proposal for creating an Acting Committee of the Scientific Board

Slide of 2019 meeting

Scientific Board Structure

- Improvements of Scientific Board structure
  - Acting Committee of the Scientific Board
    - Sub-committee of Scientific Board representatives
    - Activating the Scientific Board as level between the Executive Board and the working groups
  - Roles of Member Representatives
  - Clarification of roles of designated member representatives vs. scientific board representatives
  - Voting in Committees and Working Groups
    - A growing organization needs clear voting rules
  - Temporary working group for creating proposals
- Postpone elections (chair/vice-chair) for one year

Scientific Board Restructuring

Ingo Barkow (Scientific Board Vice-Chair)
for the temporary working group on
restructuring the Scientific Board

Recommendations for change of Scientific Board structure

- Motivation: experience shows that the current set up - a Scientific Board of approx. 40 member representatives has limited impact regarding the role of the Scientific Board in the bylaws
- After the last Scientific Board Meeting a temporary working group was setup for this purpose
- Goal: Proposal of a new Scientific Board structure
- Current status: Draft proposal which will be finalized in the next weeks (attached to the agenda of this meeting)

Scientific Board Restructuring Proposal

- The «new» Scientific Board will
  - Be comprised of 7 scientific experts
  - elected by the Designated Member Representatives of the Alliance
  - the Executive Director and the Chair of the Technical Committee will serve on the Scientific Board as ex officio members
  - external experts may be appointed by the Scientific Board for limited terms
  - Members will be elected following an Annual Meeting and serve for a term of four years except for the initial election where three will be elected for two-year terms and four for four-year terms.
  - The Chair and Vice Chair of the Scientific Board will be elected by the Scientific Board soon after the regular biennial member elections for a term of two years.

Scientific Board Restructuring Proposal

- The Scientific Community (former Scientific Board) within the Alliance comprises the scientific experts, those identified as fulfilling the scientific role.
  - Can be more involved during the year by participating in virtual meetings (instead of only at the annual meeting)

Scientific Board Restructuring Proposal

- Member Organizations are encouraged to identify individuals to fulfill 3 roles that are associated with the work of the Alliance. Note that a single person may assume more than one role:
  - Member representative related to administrative issues;
  - Scientific representative related to scientific issues;
  - Technical Contact related to technical (new).

Scientific Board Restructuring Proposal

- Tasks of the Scientific Board
  - Seek advice and consult with the Scientific Community regarding the development of products of the Alliance;
  - Make proposals to the Alliance, which votes on them as part of a work plan;
  - Set up working groups to explore new areas to support or expand the scientific interests of the Alliance according to published processes for the identification, instantiation, and conduct of working groups

Restructuring – next steps

- Finalize documents (expected within the next weeks)
- Preparation of a vote for the membership on the proposal
- If accepted –> start of nomination and election process for Scientific Board members
- Current Chair and Vice-Chair of «old» Scientific Board remain in their current positions until new Scientific Board is elected.

Scientific Board Review and Restructuring

Purpose of the Working Group

The DDI Alliance would like to improve the structure and organization of the Scientific Board, which is the scientific and technical body of the Alliance. At the 2019 Annual Meeting of the Scientific Board, a temporary working group was approved to propose a restructuring and to draft changes to the Alliance Bylaws. This work is to be completed in advance of the May 2020 annual meeting, when the finalized proposal will be discussed and voted on. The temporary working group will be chaired by Ingo Barkow, current Vice Chair of the Scientific Board.

The working group reviewed the relevant portions of the current Bylaws, Charter, DDI Scientific Process and Review, and other related documents. Documents, comments, and minutes can be found at the homepage of the Scientific Board Revision temporary working group.

https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/850853895/Scientific+Board+Revision+-+temporary+working+group

A summary of the issues which needed to be addressed by restructuring included a lack of a clear definition regarding:

- membership of the Scientific Board
- decision-making process within the Scientific Board
- identification of new areas of work, establishment of working groups

In addition, there is the desire to increase involvement of and communication with the individuals in member organizations.

The proposal encompasses the definition of new groups and redefinition of existing groups which clarify their roles and responsibilities. In particular it redefines that Scientific Board as a smaller group, elected by the Member Organizations, set up to propose specific areas of work and activity supported by the membership and to ensure the facilitation and accomplishment of the scientific workplan of the Alliance.

The Scientific Board is intended to, and needs to, be more active than the one annual meeting. It needs to act in order to improve and expand DDI semantic products and to promote their use.

Proposal Description

Definitions – the following definitions have been modified or added to Bylaws

Member Representative: An individual representing the Member Organization regarding administrative matters.

Scientific Board: The scientific and technical body of the Alliance which represents the Scientific Community. The Scientific Board proposes the scientific work plan to the membership for approval and facilitates the scientific and technical work activities.

Scientific Community: Comprises the scientific experts within the Alliance, those identified as fulfilling the scientific role.

Scientific Representative: An individual representing the Member Organization to act as contact regarding scientific matters.

Technical Contact: An individual representing the Member Organization regarding technical matters.

Proposal for Structural Change

The proposal to restructure the Scientific Board and its operations include several inter-locking ideas. These are listed here:

- Member Organizations are encouraged to identify individuals to fulfill 3 roles that are associated with the work of the Alliance. Note that a single person may assume more than one role:
    - Member representative related to administrative issues;
    - Scientific representative related to scientific issues;
    - Technical Contact related to technical.
- The Scientific Community within the Alliance comprises the scientific experts, those identified as fulfilling the scientific role.
- The Scientific Board will
    - Be comprised of 7 scientific experts, elected by the Designated Member Representatives of the Alliance; the Director and Chair of the Technical Committee will serve on the Scientific Board as ex officio members; external experts may be appointed by the Scientific Board for limited terms
    - Seek advice and consult with the Scientific Community regarding the development of products of the Alliance;
    - Make proposals to the Alliance, which votes on them as part of a work plan;

    - Set up working groups to explore new areas to support or expand the scientific interests of the Alliance according to published processes for the identification, instantiation, and conduct of working groups
- In order to support informed voting within the Alliance the Designated Member Representative is encouraged to consult with or pass the vote on each issue raised to a vote to an individual within the Member Organization with the expertise to assess the issue. In light of this each Member Organization is encouraged to specify the following positions (note that a single individual may serve in more than one capacity):
    - Member Representative on administrative;
    - Scientific representative on scientific issues;
    - Technical contact on technical issues.

The base of the proposal is to make changes in the Bylaws in the description of the Scientific Board in the following areas:

A. Purpose

Discussion and consensus on the purpose is summarized on the homepage of the Scientific Board Revision temporary working group.

https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/850657321/Gathering+Views+On+By-laws+-+Purpose.

Further discussions also took place, which intersected with bylaws relating to organization, voting, and elections.

The outcome of the consensus agreement is reflected in the proposed changes to the Bylaws. In short, this led to a new clause in the Purpose section to emphasize the strategic and coordinating role of the Scientific Board. As a result of it being reduced in size, the decision-making role has been moved explicitly to the Designated Member Representatives and the voting clause has been removed, along with the evaluation of new proposals for standards and work productions.

The wording standards and semantic products have been revised in line with the current description of the DDI Alliance outputs, "DDI standards and other work products". Detailed changes to the Bylaws are found in “Bylaws 2020 draft”. In summary these include changes in the following areas:

- Section II.A changed definition of Scientific Board and added:
    - Scientific Community
    - Member Representative
    - Scientific Representative
    - Technical Contact

- Section VI.B.1.a (a) and (d) minor wording changes
- Section VI.B.2.a and b minor wording changes
- Section VII.A. replaced description of purpose
- Section VII.B. replaced description of organization
- Section VII.C. Elections (added) (renumbering of remaining sub-parts)

The ambition is that these changes to the bylaws will engender a smaller, more flexible strategic and co-ordination body that provides an annual report to the membership The report would be expected to also include a forward plan / roadmap, which would be approved by the membership and would be able to be actioned through the following year.

When a decision of the Annual Meeting has been reached, appropriate changes to the Standards Development and Review Process and Procedures document will be submitted.

Proposed Revision to the Bylaws Section VII.A.

The purposes of the Scientific Board are to:

1. Provide direction and coordination in the development of the substantive content of the DDI standards and other work products of the Alliance by its sub-committees and working groups within the context of the Alliance Strategic Plan
2. Oversee the substantive content of DDI standards and other work products
3. Undertake research and testing concerning proposals for DDI standards and other work products
4. Develop and promulgate best practices for use of DDI standards and work products
5. Assess progress and barriers to progress
6. Provide a report on progress of the scientific program over the previous year, and proposals for the future scientific direction and related activities to the Annual Meeting of the Alliance.
7. To enact the scientific program agreed at the Annual Meeting

B. Organization

Following the revision of the purposes of the Scientific board, the section VII.B Scientific Board: Organization must be amended to accommodate the change in the size and organization of the Scientific Board. By consensus of the working group the Scientific Board should be comprised of 3 member types:

- Elected members: members elected by the Designated Representatives of the Member Organizations of the Alliance. These members have voting rights within the functioning of the Scientific Board such in the selection of Chair and Vice-Chair, or in determining proposals to put before the membership. Any individual from a Member Organization or Associate Member Organization may run for an elected position on the Scientific Board.

- Ex officio members: The Executive Director and the Chair of the Technical Committee serve as ex officio members. The Technical Committee is a standing sub-committee of the Scientific Board and close coordination is needed.
- Advisory members: Non-voting members who represent external groups

The Scientific Board would be composed of 7 elected members, 2 ex officio members, and up to 2 advisory members for a total size of 9-11 persons. This format allows for more regular meetings of the Scientific Board, sufficient size to ensure diversity and allow for the smooth rotation of members over time without major interruption of work.

Provisions are included to guarantee rotation in membership of the Scientific Board as well as separation in the positions of officers of the different boards of the Alliance.

The reduction in size will facilitate the Scientific Board in identifying and proposing scientific directions for the approval of the membership.

The Technical Committee and other sub-committees and working groups continue to act within the strategic directions proposed by the Scientific Board and agreed to by the membership.

The Scientific Board will have to report to the Annual meeting of the Alliance.

Proposed Revision to the Bylaws Section VII.B.

1. The Scientific Board shall be composed of seven members elected by the Members of the Alliance.
2. The Executive Director and Chair of the Technical Committee shall serve as ex-officio members, without internal vote, of the Scientific Board.
3. The Scientific Board may appoint up to two external Advisory Members, without internal vote.
4. Representatives from Members and Associate Members of the Alliance are eligible to serve as elected members of the Scientific Board.
5. No Member or Associate Member shall have more than one representative serving on the Scientific Board at the same time.
6. Elected members of the Scientific Board shall serve no more than four consecutive terms.
7. The Chair and Vice-Chair of the Scientific Board are determined by the elected members of the Scientific Board and shall serve no more than three consecutive terms.
8. Only elected members of the Scientific Board are eligible to serve as Chair or Vice-Chair.
9. No officer (chair or vice-chair) of the Executive Board nor of the Technical Committee may serve as an officer of the Scientific Board.
10. The Technical Committee is established as a standing Sub-Committee of the Scientific Board.

11. The Scientific Board may establish Sub-Committees and Working Groups on specific topics.
12. The Scientific Board shall have oversight of every Sub-Committee and Working Group established under it.
13. Other meetings before the Alliance may be called by the Scientific Board as needed.
14. A quorum shall consist of 4 elected members of the Scientific Board.
15. The Scientific Board should maintain a document outlining its internal operational procedures entitled “Scientific Board Operational Guidelines”.

C. Elections

The creation of a smaller Scientific Board with elected membership requires the addition of wording to the Bylaws to govern the election of members.

Proposed Revision to the Bylaws Section VII.C (added)

1. Members will be elected following an Annual Meeting and serve for a term of four years except for the initial election where three will be elected for two-year terms and four for four-year terms.
2. Terms will start on July 1 of the election year.
3. Any member vacancy will be filled by election as soon as possible and that member will begin serving when elected for the remainder of the vacating member's term.
4. In election years, nominations for members will be solicited in April and a slate will be prepared by the Executive Director for discussion at the Annual Meeting with the election occurring in June. In the event, that there are more candidates than positions, the election will be decided on the basis of those candidates getting the most votes. If a tie vote occurs, a second round of voting will take place.
5. The Chair and Vice Chair of the Scientific Board will be elected by the Scientific Board soon after the regular biennial member elections for a term of two years.

Scientific Board Operational Guidelines

It is not the intent of the working group to dictate the inner workings of the Scientific Board outside of its description in the Bylaws. However, its workings, particularly in terms of the creation of working groups and facilitation of the scientific and technical work of the Alliance should be transparent. Therefore, the Scientific Board’s first task should be the creation of a document outlining the operational procedures of the Scientific Board. It should include the following sections:

Definitions

Add definitions as required for the understanding of this document

A. Purpose

Clarify specific aims and goals of the Scientific Board within the structure provided in the Bylaws

B. Organization
Internal organization
Details of specific procedures such as the disposition of working group proposals

C. Elections
Internal voting procedures for Chair and Vice-Chair within the structure provided in the Bylaws
Internal decision-making process for other decision making areas

E. Roles
The working group recommends the following content:

The Bylaw amendments and guidelines were driven by the goals of both broadening the expert scientific and technical input and participation of Alliance members and others, as well as ensuring accountability by the Scientific Board to the Alliance The following are recommendations and guidelines therefore non-binding and so have not been added to the Bylaws. Nevertheless, these recommendations and guidelines serve as a base for upcoming processes and should be incorporated into the organizational guidelines document of the Scientific Board when established.

1. Member organizations will be asked to nominate one or more persons to the following roles - Member Representative, Scientific Representative (as before) and Technical Contact (new).
2. The Member Representative serves as the primary contact to the DDI Alliance in general and especially the Executive Board for organizational questions.
3. The Scientific Representative serves as the contact person for the Scientific Board and related workgroups especially for questions on requirements and future directions.
4. The Technical Contact person is a new role which serves as the contact for the Technical Committee to give inputs to technical implementations of the standard
5. All of these roles are invited to the annual meeting where the Executive Board and Scientific Board report to the whole Alliance.
6. The Technical Committee, as a permanent standing committee, reports as a part of the Scientific Board report.
7. During the year all of these roles can be called upon as needed to provide input, participate in virtual meetings, or become involved in working groups
8. Scientific Contacts within the Member Agency should be designated to provide support and/or input to the Scientific activities of the Alliance. Scientific Contacts should express knowledge in one or more of the following areas:
    a. expert understanding of metadata and what it can do
    b. working knowledge of statistical lifecycle and survey methodology
    c. knowledge of future directions in statistical analysis
    d. working knowledge of experimental design
    e. working knowledge of some other (outside social science) data domains
    f. understanding of new data representation techniques

    g. knowledge of data infrastructures
9. Technical Contacts within the Member Agency should be designated to provide support and/or input to the technical activities of the Alliance in the following areas:
    a. Implementation of DDI and other metadata management on a technical level
    b. Modeling in various representations (UML, XML, RDF, JSON, etc.)
    c. Technological changes in data and metadata storage and access
    d. Preparation of technical documentation for implementers of DDI
    e. Technical testing of DDI products and development work

Scientific Board Operational Guidelines

The Scientific Board should prepare operational guidelines in the following areas for the purpose transparency and codifying any normalized processes of the Scientific Board

Definitions
Add definitions as required for the understanding of this document

A. Purpose
Clarify specific aims and goals of the Scientific Board within the structure provided in the Bylaws

B. Organization
Internal organization
Details of specific procedures such as the disposition of working group proposals

C. Elections
Internal voting procedures for Chair and Vice-Chair within the structure provided in the Bylaws
Internal decision making process for other decision making areas

E. Roles

Suggested for inclusion:

1. Member organizations will be asked to nominate one or more persons to the following roles - Member Representative, Scientific Representative (as before) and Technical Contact (new).
2. The Member Representative serves as the primary contact to the DDI Alliance in general and especially the Executive Board for organizational questions.
3. The Scientific Representative serves as the contact person for the Scientific Board and related workgroups especially for questions on requirements and future directions.
4. The Technical Contact person is a new role which serves as the contact for the Technical Committee to give inputs to technical implementations of the standard
5. All of these roles are invited to the annual meeting where the Executive Board and Scientific Board report to the whole Alliance.
6. The Technical Committee, as a permanent standing committee, reports as a part of the Scientific Board report.
7. During the year all of these roles can be called upon by their respective higher level group (Executive Board, Scientific Board, Technical Committee) to participate in virtual meetings or invitations to working groups

8. Scientific Representative within the Member Agency should be designated to provide support and/or input to the Scientific activities of the Alliance. Scientific Contacts should express knowledge in one or more of the following areas:
    a. expert understanding of metadata and what it can do
    b. working knowledge of statistical life-cycle and survey methodology
    c. knowledge of future directions in statistical analysis
    d. working knowledge of experimental design
    e. working knowledge of some other (outside social science) data domains
    f. understanding of new data representation techniques
    g. knowledge of data infrastructures
9. Technical Contacts within the Member Agency should be designated to provide support and/or input to the technical activities of the Alliance in the following areas:
    a. Implementation of DDI and other metadata management on a technical level
    b. Modeling in various representations (UML, XML, RDF, JSON, etc.)
    c. Technological changes in data and metadata storage and access
    d. Preparation of technical documentation for implementers of DDI
    e. Technical testing of DDI products and development work

I. Preamble

As described in the Charter, the Alliance is an unincorporated, self-sustaining membership organization whose members have a voice in the development, promotion, and dissemination of DDI specifications.

II. Definitions

Annual Meeting of Members: An assembly of Member Representatives convened for the Alliance's annual business meeting.

Associate Member Organization: A Member Organization that does not pay dues.

Chair: The individual elected by the Designated Member Representatives to lead Annual Meetings and to serve as Chair of the Executive Board.

Designated Member Representative: An individual designated by the Member Organization to exercise its voting rights.

Executive Board: The policy-making and oversight body of the Alliance.

Executive Director: Individual from the Host Institution who runs the Secretariat and coordinates Alliance activities.

Executive Director's Advisory Group: Small group that advises the Executive Director on an ongoing basis. Good Standing: Payment of annual Alliance dues (if applicable), provision of in-kind contributions, and adoption of DDI standards and products as appropriate.

Host Institution: An organization providing an operational base and assuming direct financial and legal responsibility for the Alliance.

Member Organization: An organization that is in good standing in the Alliance.

Member Representative: An individual representing the Member Organization regarding administrative matters.~~An individual appointed by the Member Organization to represent it at the Annual Meeting of Members and other Alliance meetings.~~

Observers: Individuals from Member Organizations who are not Member Representatives, or individuals from non-member organizations, participating in meetings. Observers participate without voting rights.

Scientific Board: The scientific and technical body of the Alliance which represents the Scientific Community. The Scientific Board proposes the scientific work plan to the membership for approval and facilitates the scientific and technical work activities.

Scientific Community: Comprises the scientific experts within the Alliance, those identified as fulfilling the scientific role.

Scientific Representative: An individual representing the Member Organization to act as contact regarding scientific matters.

Secretariat: The administrative arm of the Alliance that provides financial and clerical support.

Sub-Committee: A subset of Member Representatives that is established by either the Member Representatives or the Scientific Board and created by formal resolution for a defined purpose or objective and for a specified period of time.

Technical Committee (TC): Scientific Board standing committee that models, renders, maintains, and updates the specifications.

Technical Contact: An individual representing the Member Organization regarding technical matters.

Vice Chair: The individual elected by the Designated Member Representatives to lead Annual Meetings in the absence of the Chair and to serve as Vice Chair of the Executive Board.

Working Group: A group composed of Member Representatives and possibly individuals outside Member Organizations that is established by formal resolution of the Scientific Board or the Executive Board, for a defined purpose or objective and for a specified period of time.

III. Purposes

The Alliance is made up of diverse organizations from a range of countries, disciplines, and sectors committed to developing and maintaining publicly available metadata standards and semantic products for documenting social science and related data. The Alliance's purposes are to further its Mission and to fulfill the Objectives in the Alliance Charter.

IV. Organization

An Executive Director and an Executive Board manage the operations of the Alliance, while a Scientific Board is responsible for its scientific and technical work. A small Secretariat is maintained to administer day-to-day operations. The Alliance is financially self-supporting through a variety of revenue streams, including membership dues; license fees; workshops, symposia, and publication fees; and external research or training grants and contracts with the Host or a Member Institution.

The Alliance is a Program of the University of Michigan (UM) as the current Host Institution and operates within the Inter-university Consortium for Political and Social Research (ICPSR) in accordance with the Alliance's Charter and Bylaws and with the policies and regulations of the University of Michigan.

V. Products

The DDI standards and semantic products are advanced through development lines. These development lines and products, which shall be publicly available, are described in a standing document, "Standards Development and Review Process and Procedure".

VI. Membership

A. Terms of Membership and Dues

1. Membership in the Alliance is open to any organization or agency that maintains its good standing, has a material interest in the work of the Alliance, agrees to the terms of the Membership Agreement, and is in compliance with this Charter and Bylaws. All Members shall have the same rights, although the Executive Board may create differing classifications of

membership for the purpose of levying annual dues. Multiple memberships from a single agency, organization, or institution are admitted on a case-by-case basis subject to the approval by formal resolution of the Executive Board. Where multiple memberships are granted, each membership shall appoint a Member Representative with the rights and obligations described in this section.

2. The Executive Board shall establish a schedule for payment of annual dues for each classification of membership, payable in U.S. dollars. This payment shall be made in full upon receipt of the annual invoice. Not more than once per year, the Executive Board may modify the classification of membership and raise or lower the annual dues after consultation with and ratification by the Members at the Annual Meeting. No Member Organization may have its dues raised until the end of its yearly membership period. At the discretion of the Executive Board, annual dues may be reduced for Members located in developing countries or in countries whose economies are in transition. Such reduction of membership dues will be reported annually to and ratified by the Members at the Annual Meeting.

B. Rights and Obligations of Members

1. A Member Organization:

(a) Shall have a seat at the Annual Meetings of Members and the Scientific Board of the Alliance. Its Designated Member Representative should attend the Annual Meeting and may also serve on the Scientific Board ~~as the liaison to the Member Organization~~.

(b) Shall have one vote exercisable on its behalf by its Designated Member Representative. A Member Organization shall provide the Executive Director with the name of its Designated Member Representative prior to any vote.

(c) Shall be eligible to have a Member Representative elected to the Executive Board with all rights and privileges of a member of the Executive Board.

(d) May send Observers to the Annual Meetings of Members and the Scientific Board of the Alliance, subject to space limitations.

(e) Should participate in at least one substantive activity of the Alliance, such as an elected position, Technical Committee, Sub-Committee, or Working Group through its Member Representative or other employees.

(f) May display the Alliance trademarks on promotional material and publicize the Member's participation in the Alliance.

(g) May send a specified number of participants, as determined by the Executive Board, to selected Alliance-sponsored workshops and symposia without payment of workshop and symposia fees.

(h) May request access to Member-only information for employees of its organization.

(i) The Member Organization should provide a Member Representative, Scientific Representative, and Technical Contact to support communication between the Member

Organization and the Alliance. A single individual could fulfill multiple roles. These individuals should attend the appropriate meetings of the Alliance.

2. An Associate Member Organization:

(a) Shall have a seat at the Annual Meetings of Members and the Scientific Board of the Alliance.

(b) May send Observers to the Annual Meetings of Members and the Scientific Board of the Alliance, subject to space limitations.

(c) Should participate on at least one substantive activity of the Alliance, such as an elected position, Technical Committee, Sub-Committee, or Working Group through its Member Representative or other employees.

(d) May display the Alliance trademarks on promotional material and publicize the Member's participation in the Alliance.

(e) May request access to Member-only information for employees of its organization.

(f) The Associate Member Organization should provide a Member Representative, Scientific Representative, and Technical Contact to support communication between the Member Organization and the Alliance. A single individual could fulfill multiple roles. These individuals should attend the appropriate meetings of the Alliance.

3. If the Member Organization is itself a consortium, user society, professional association, or otherwise has members or sponsors, the rights and privileges granted under Alliance membership extend only to the paid employees or designated representatives of the Member Organization, not to such organization's individual members or sponsors.

VII. Scientific Board

A. Purpose

The purposes of the Scientific Board are to:

1. Provide direction and coordination in the development of the substantive content of the DDI standards and other work products of the Alliance by its sub-committees and working groups within the context of the Alliance Strategic Plan
2. Oversee the substantive content of DDI standards and other work products
3. Undertake research and testing concerning proposals for DDI standards and other work products
4. Develop and promulgate best practices for use of DDI standards and work products
5. Assess progress and barriers to progress
6. Provide a report on progress of the scientific program over the previous year, and proposals for the future scientific direction and related activities to the Annual Meeting of the Alliance.
7. To enact the scientific program agreed at the Annual Meeting

~~1. Contribute to the substantive content of DDI standards and semantic products and approve major version revisions.~~

~~2. Evaluate technical proposals through the Alliance standards review process.~~

~~3. Undertake research and testing concerning proposals for DDI standards and semantic products.~~

~~4. Develop and promulgate best practices for use of DDI standards and semantic products.~~

~~5. Assess progress and barriers to progress.~~

~~6. Suggest future directions and activities for the Alliance.~~

B. Organization

1. The Scientific Board shall be composed of seven members elected by the Members of the Alliance.
2. The Executive Director and Chair of the Technical Committee shall serve as ex-officio members, without internal vote, of the Scientific Board.
3. The Scientific Board may appoint up to two external Advisory Members, without internal vote.
4. Representatives from Members and Associate Members of the Alliance are eligible to serve as elected members of the Scientific Board.
5. No Member or Associate Member shall have more than one representative serving on the Scientific Board at the same time.
6. Elected members of the Scientific Board shall serve no more than four consecutive terms.
7. The Chair and Vice-Chair of the Scientific Board are determined by the elected members of the Scientific Board and shall serve no more than three consecutive terms.
8. Only elected members of the Scientific Board are eligible to serve as Chair or Vice-Chair.
9. No officer (chair or vice-chair) of the Executive Board nor of the Technical Committee may serve as an officer of the Scientific Board.
10. The Technical Committee is established as a standing Sub-Committee of the Scientific Board.
11. The Scientific Board may establish Sub-Committees and Working Groups on specific topics.
12. The Scientific Board shall have oversight of every Sub-Committee and Working Group established under it.
13. Other meetings before the Alliance may be called by the Scientific Board as needed.
14. A quorum shall consist of 4 elected members of the Scientific Board.
15. The Scientific Board should maintain a document outlining its internal operational procedures entitled “Scientific Board Operational Guidelines”.

~~1. The Scientific Board shall be composed of Member and Associate Member Organization Designated Representatives and shall be staffed by the Secretariat.~~

~~2. The Executive Director is an ex-officio member, without vote, of the Scientific Board. The Executive Director is not eligible to serve as Chair or Vice-Chair.~~

~~3. At least one-third of the Members present at a meeting of the Scientific Board properly called by the Executive Director shall constitute a quorum.~~

~~4. The Executive Director may invite others to participate as Observers in activities and meetings of the Scientific Board.~~

C. Election

1. Members will be elected following an Annual Meeting and serve for a term of four years except for the initial election where three will be elected for two-year terms and four for four-year terms.
2. Terms will start on July 1 of the election year.
3. Any member vacancy will be filled by election as soon as possible and that member will begin serving when elected for the remainder of the vacating member's term.
4. In election years, nominations for members will be solicited in April and a slate will be prepared by the Executive Director for discussion at the annual meeting with the election occurring in June. In the event that there are more candidates than positions, the election will be decided on the basis of those candidates getting the most votes. If a tie vote occurs, a second round of voting will take place.
5. The Chair and Vice Chair of the Scientific Board will be elected by the Scientific Board soon after the regular biennial member elections for a term of two years.

D~~C~~. Technical Committee

1. The purpose of the Technical Committee is to model, render, maintain, and update the DDI specifications to meet community needs and align with Alliance strategic goals. The TC receives input from substantive working groups of the Scientific Board, DDI users and developers, and other interested parties.
2. The activities of the Technical Committee cover the following:
    a. Develop the conceptual models.
    b. Implement the models in various technical forms.
    c. Monitor the metadata landscape and related developments.
    d. Initiate and plan possible future directions for the standards.
3. The Technical Committee will elect a Chair and Vice Chair for a three-year term. The Chair and Vice-Chair are eligible for re-election.

VIII. Executive Board

A. Purpose

The purposes of the Executive Board are to:

1. Select a Host Institution to house the Executive Director and Secretariat and to assume financial and legal responsibility for the Alliance.

2. Set overall policy and budget for the Alliance.

3. Provide strategic guidance and review of the Alliance's activities.

4. Appoint an Executive Director for the Alliance for a five-year, renewable term.

5. Oversee the management of the financial affairs of the Alliance on behalf of the Members.

6. Set Alliance membership fees and length of membership term for each institutional classification of membership subject to the ratification by formal resolution at the Annual Meeting of Members.

7. Set fees for use of Alliance registered trademarks, certification marks, and collective marks or copyright material and for Alliance-sponsored activities and products.

8. Make decisions on allocation of funds for innovative work and testing.

9. Form Working Groups to perform specific duties.

B. Organization

1. The Executive Board is composed of seven voting members: six At-Large members elected by the Designated Member Representatives and one member appointed by the Host Institution.

2. The Executive Director shall serve as an ex-officio member, without vote, of the Executive Board.

C. Elections

1. At-Large members will be elected following an Annual Meeting and serve for a term of four years except for the initial election where half will be elected for two-year terms and half for four-year terms. Terms will start on July 1 of the election year. Any At-Large member vacancy will be filled by election as soon as possible and that member will begin serving when elected for the remainder of the vacating member's term.

2. In election years, nominations for At-Large members will be solicited in April and a slate will be prepared by the Executive Director for discussion at the annual meeting with the election occurring in June. In the event that there are more candidates than positions, the election will be decided on the basis of those candidates getting the most votes. If a tie vote occurs, a second round of voting will take place.

3. The Chair and Vice Chair of the Executive Board will be elected by the Board soon after the regular biennial At-Large member elections for a term of two years and each may serve no more than three consecutive terms. The Chair and Vice Chair of the Executive Board will also serve as the Chair and Vice Chair of subsequent Annual Meetings of the Membership Representatives.

IX. Executive Director

The role of the Executive Director is inter alia to:

1. Supervise the Secretariat and prioritize its work.

2. Convene the Annual Meeting of Members, meetings of the Executive Board, and the Scientific Board.

3. Maintain the list of Designated Member Representatives of Member Organizations.

4. Represent the Alliance at conferences, meetings, and other forums, or designate someone from the Alliance to do so.

5. Make programmatic decisions based upon the recommendations of the appropriate committees.

6. Coordinate Alliance activities around the world.

7. Assist the Host Institution and Members to raise funds for DDI-related activities.

8. Present an annual Financial Report to the Executive Board.

9. Present an annual Activities Report to the Alliance.

10. Designate Member Representatives as official Alliance representatives to other organizations and committees.

As appropriate, the other members of the Executive Board will assist the Executive Director in these activities. The Executive Director may also appoint appropriate staff as needed.

The Executive Director serves as an ex officio member of the Executive Board and the Scientific Board, without vote.

The Executive Board will negotiate an agreement with the Host Institution to house the Executive Director and the Secretariat and to assume financial and legal responsibility for the Alliance. The work of the Secretariat will be supported by membership dues and other fees. The tasks of the Secretariat shall include the following:

1. Develop and manage a Web site for communication within the Alliance and with the public.

2. Arrange for and facilitate meetings of the Member Representatives, Executive Board, and the Scientific Board.

3. Support the Executive Director's work.

4. Arrange for and facilitate any Alliance elections and votes.

5. Publish such material as is directed by the Member Representatives, the Board of Experts, or the Executive Board.

6. Provide for the ongoing functioning of the DDI Agency Registry.

7. Organize workshops as directed by the Scientific Board.

8. Collect dues and fees.

9. Maintain auditable financial records and accounts.

10. Produce annually a Financial Report detailing income and expenditures for review by the Executive Board and circulation to the Member Representatives.

11. Produce annually an Activities Report for the Member Representatives, Board of Experts, and Executive Board.

12. Solicit additional Member Organizations to join the Alliance.

13. Conduct such other business as assigned by the Executive Board.

X. Executive Director's Advisory Group

A. Purpose

The purpose of the Advisory Group is to advise the Executive Director on an ongoing basis.

B. Organization

The Advisory Group shall be composed of the Chairs and Vice Chairs of the Executive Board, the Scientific Board, and the Technical Committee.

XII. Working Groups

A. Purpose

Working Groups advise the Executive Board or the Scientific Board on relevant topics and activities related to the operation, development and future of the Alliance and its specifications and semantic products.

B. Organization

1. Working Groups may be created by the Executive Board or the Scientific Board.

2. Membership is drawn from the Member Representatives and such other persons as may be appointed by the body creating the Working Group. In general, Working Groups should broadly represent the community with relevant knowledge and expertise about the subject area that is the focus of the Group's work.

3. A Working Group shall have a designated leader who is responsible for managing the work of the group and reporting on an annual basis to its authorizing body.

XIII. Meetings

A. Official Meetings

The Alliance has a number of possible meetings, which may or may not occur during the course of a calendar year. These include, but are not limited to:

- Annual Meeting of Member Representatives.

- Meeting of the Scientific Board.

- Meeting of the Executive Board.

B. Annual Meeting of Member Representatives

1. The Annual Meeting of Member Representatives occurs once within a calendar year.

2. The purposes of the Annual Meeting of Member Representatives are to:

(a) Provide a forum for Member Organization discussion and feedback.

(b) Review and approve the activities of the Executive Board in the preceding year.

(c) Receive and approve by formal resolution the Annual Report of the Executive Director.

(d) Deliberate any proposals to amend the Charter and Bylaws.

3. The Annual Meeting of Member Representatives shall be called separately by the Executive Director and may precede or follow a meeting of the Scientific Board.

4. The Annual Meeting of Member Representatives shall be chaired by the Chair of the Executive Board.

5. At least one-third of the Designated Member Representatives present at an Annual Meeting properly called by the Executive Director shall constitute a quorum.

6. The Member Representatives may meet more often than annually if called to do so

(a) By formal resolution of the Executive Board.

(b) By presentation of a petition to the Executive Board drafted for that purpose and approved by one-third of the Designated Member Representatives.

C. Meeting of the Scientific Board

1. The Scientific Board shall meet at least once per year.

~~2. At least one-third of the members of the Scientific Board present at a meeting properly called by the Executive Director shall constitute a quorum.~~

D. Meeting of the Executive Board

1. The Executive Board shall meet at least once per year to discuss matters related to its purpose.

2. At least one-third of the members of the Designated Member Representatives ~~Scientific Board~~ present at a meeting properly called by the Executive Director shall constitute a quorum.

XIV. Budget

The Executive Board shall establish a budget that provides financial support for the successful operation of the Alliance that may include support for some portion of the time of the Executive Director, Alliance duties and functions as determined by the Executive Director and the Secretariat, expert consultation, meetings, training, and funds for innovation and testing.

XV. Specification Review Process and Procedure

Every proposal for a modification to an existing specification goes through a standard review process, unless an alternative process is later approved by the Executive Director and the Scientific Board. The standard review process is documented in “Standards Development and Review Process and Procedure.”

XVI. Visiting Experts

Member Organizations may volunteer to contribute staff on assignment to the Host Institution or Member Organizations for specific implementation efforts sponsored by the Alliance. If the Host Institution or Member Organization has the resources to accept such staff, the visitors will be provided with appointments as Visiting Experts. For the portion of their time assigned to Alliance activities, visitors will coordinate their work with the Executive Director based on Alliance priorities.

XVII. DDI Standard Publicly and Internationally Available

The DDI standards shall be publicly and internationally available free of charge to any one.

XVIII. Intellectual Property

As the current Host Institution, the University of Michigan, on behalf of the Alliance, will maintain, protect and license all registered trademarks, certification marks and collective marks or copyright held by it, or held in the name of the Alliance or on behalf of the Alliance, by the University of Michigan, solely in accordance with these Bylaws. Costs associated with these activities will be borne by the Members through assessment of dues and other fees as necessary.

XIX. Amendments to the Charter and Bylaws

Any Member Organization may propose an amendment to the Charter and Bylaws by drafting a petition to be signed by at least one-third of the Member Organizations. Amendments may also be proposed by a simple majority of the Executive Board. Proposals to amend the Charter and Bylaws shall be deliberated at the Annual Meeting of the Member Representatives.

Amendments must be adopted by a two-thirds majority vote of the Designated Member Representatives after written electronic notice of the vote of at least sixty days. No amendment may void the condition of the Bylaws that DDI standards shall be publicly and internationally available free of charge, whether or not that organization is a Member of the Alliance (Section XVII).

DDI – Cross Domain Integration (DDI-CDI)

Arofan Gregory (MRT Group moderator)
for the MRT - Modeling, Representation, and Testing Lifecycle Working Group
(CDI development group)

DDI – Cross Domain Integration (DDI-CDI)

Feature Summary and Status

DDI Alliance Scientific Board

18 May, 2020

Outline

- Background: MRT and DDI 4 Core
- Group and Events
- DDI-CDI within the DDI Product Suite
- DDI-CDI Features
- DDI-CDI Alignment with Other Standards
- Current Status/Timelines

MRT and DDI 4 Core

- In the margins of the 2018 EDDI meeting (Berlin) it was agreed that a “core” of the DDI 4 work should be brought to market
- A 1-year timeframe was proposed
- The Modelling, Representation and Testing (MRT) group was formed in early 2019
- The working process was to base models on implementations, tested against real-world use cases
  - ALPHA Network
  - DDI R Libraries
  - Others (BLS for time series, etc.)

Group and Events

- Small group (9 members) meeting weekly (and more) for over a year
- No turn-over – members have been extremely focused and disciplined
- Ottawa Sprint in margins of NADDI 2019
- Dagstuhl Sprint in October 2019
- Public Review Release April 2020
- Communications with management, technical committee work, marketing, and training have been emphasized

Evolution in Purpose

- DDI-CDI was expected to be the “core” of a model-driven DDI
  - A “next generation” after DDI-Lifecycle
- Implementation cases showed that something else was needed: a focus on data provenance and data integration
- DDI-CDI has emerged as a *companion* to DDI-Codebook and DDI-Lifecycle, not a replacement for them
- The SBE community needs better data integration tools
  - So do other domains!

Real-World Trends and Requirements

- Several changes have taken place in recent years which impact the requirements for DDI-CDI
  - Larger research projects using data sometimes coming from external domains
  - **More** data, coming from a wider range of sources
  - Increased ability to compute with data (Machine Learning, etc.)
- These changes result in requirements for data/metadata management
  - More complete, machine-actionable metadata is needed
  - Improved “context” for data is needed (provenance, semantics)
  - New data formats/structures must be described and integrated
  - A broader range of technology platforms require support

DDI-CDI within the Product Suite

- DDI-CDI does not replace DDI-C or DDI-L
  - It can and will be used in combination with other DDI specifications
- It adds support for describing new types of data
- It expands the ability to describe process/provenance
- It provides a detailed description of integration between disparate types of data
- Extends the applicability of the DDI to new domains/disciplines
  - As an integration tool
  - As a data management tool

DDI-CDI Functionality

- Describe data formats:
  - Rectangular/unit-record
  - Long/event
  - No-SQL/”big data”
  - Multi-dimensional
- Describe data provenance/process
  - Procedural process
  - Declarative process
- Describe “foundational” metadata
  - Codes/categories/classifications
  - Concepts, variables, etc.

Design Goals

- Produce a useful, implementable product based on real use cases
- Produce a standard which would be useful across technology platforms (model-driven)
- Produce a standard which is more approachable and easier to understand
  - W3C specifications used as a model
  - Lots of examples at different levels

Data Description

- Focus is on the role played by individual datums across different types of structures
  - The same data point performs different functions (measure, descriptor, identifier) in different data sets
- We can describe 4 types of data structure
  - The model can easily extend to describe others
- Data transformation tools perform this kind of thing all the time
  - DDI-CDI can express the relevant metadata for tracking datums across different structures
  - No other standard has this capability

Process and Provenance

- DDI has never attempted to describe the processes which are combined to actually produce data
  - Focus has always been on low-level data processing (stats packages/SDTL)
- DDI-CDI describes processes at a higher level, and connects them with low-level processing descriptions
- Directly implements common models for provenance and process (PROV, BPMN)
- Supports “black box” parallel processing as well as stepwise “flow” processing
  - New feature of DDI
  - Becoming common in the real world

Foundational Metadata

- Building on years of work in DDI 4
- Sophisticated model for variables, conceptual underpinnings/application
- Works flexibly with different ontologies/concept systems/thesauri
- Well-aligned with DDI-Lifecycle

A Model-Driven Standard

- DDI-CDI uses a UML formalization
  - Allows for use in a wide range of design and development environments
  - Supports more explicit standards alignment
  - Limited subset of UML features
- The “official” expression of the model is in Canonical XMI
  - An XML-based exchange format for UML models
  - Profiled to work with the widest possible range of UML tools
- Enhances clarity of the model and “future-proofs” it
- Provides platform-independence

DDI-CDI Alignment with Other Standards

- DDI-CDI directly implements other standards at the level of UML ("trace" relationships)
- DDI-CDI is *domain-neutral*
- Aligned with other flavors of DDI (Codebook. Lifecycle, etc.)
- Directly implements process/provenance standards (BPMN, PROV)
- Supports GSIM/GSBPM
- Designed to integrate with discovery standards (Schema.org, DCAT)
- Aligned with other data description models (CSV on the Web, SDMX, DataCube, Observable Properties, SOSA/SSN, etc.)
  - Some work remains in testing these alignments

Current Status/Timeline

- Public review period ongoing through July 2020
- Series of webinars to recruit meaningful review from other domains
  - CODATA is supporting this activity
- Revised review version released in September 2020
- Focused review at intensive Dagstughl workshop or virtual equivalent
  - CODATA has offered to convene a working group of reviewers from external domains to feed requirements into MRT
- First production release early 2021

Questions?

MRT Report: DDI – Cross Domain Integration (DDI-CDI) Features and Status

12 May 2020

I. Introduction and Background

This document summarizes progress-to-date by the Modelling, Representation, and Testing (MRT) working group, and provides a description of features and status for the DDI-CDI specification. It also lays out the intent of the design, and the working process which produced the current draft of the specification.

After several years of work on the DDI 4/Moving Forward next-generation DDI model, it was felt that something must be brought to market which demonstrated its utility in practical terms. The Prototype Review had revealed several important aspects of the modeling work but was not itself implementable as a finished product.

In the margins of the 2018 EDDI meeting in Berlin, a decision was made to launch an effort to identify and implement a core set of the DDI 4 features in a project which would be of a year's duration. In early 2019, the Modeling, Representation, and Testing working group was formed to perform this task. The group consisted of nine individuals involved in the DDI 4 work, and it has met on a weekly (and sometimes more-than-weekly) basis since the formation of the group. The pace of the work has been aggressive.

The working process of the group was based on Agile approaches, as had been the earlier DDI 4 Sprints, but stronger focus was placed on having real-world use cases to drive the requirements, and to include the generation of syntax representations and to test them through implementation.

While not always working as smoothly as hoped, this approach was the one followed, resulting in the identification of several relevant use cases, and the testing of the core model to support needed features of those projects. These included the following:

- **The ALPHA Network:** a network of Health and Demography Surveillance Sites in eastern and southern Africa, using DDI Codebook to document local surveys which were then compiled, cleaned and integrated to form a single unified research dataset housed at the London School of Hygiene and Tropical Medicine. An emphasis was placed on the documentation of the process by which the local survey data were transformed into the series of events used for analysis. Process metadata was programmatically harvested from the transformation platform and enriched by those performing the work.
- **DDI R Libraries:** Lead by Larry Hoyle at University of Kansas, this project was a prototype implementing a series of useful functions in the popular R Statistical package across different versions of DDI-CDI, including the integration of different types of data used for analysis and manipulation of relevant metadata.

- **Bureau of Labor Statistics Time Series Model:** The BLS is developing a unified model for the time series in its existing systems. While not using the representation of DDI-CDI, this was a practical test case for the model, to determine its expressive capabilities vis-à-vis an existing implementation for multi-dimensional data.
- **Others:** Several other examples were used at points during the work, including the Microdata.no system for secure online access to Norwegian register data, and various implementations at Statistics Canada. These were selected to test specific aspects of the design (e.g., transformation across different data structures for storage/retrieval and analysis, big data systems, declarative process management).

The major design goals for DDI-CDI emerged from these cases. New forms of data would need to be described which were not sufficiently covered by existing DDI specifications (NoSQL/big data, event data, multi-dimensional data). The description of data provenance, notably from a process perspective, was needed. Throughout, some of the strengths of the DDI 4 model were emphasized: the datum-oriented description of data, the "variable cascade" for describing how different types of variables are related, the mature model for describing classifications, and the documentation of the use of concepts in different parts of the metadata, among others.

II. Main Features of DDI-CDI

DDI 4 was originally conceived as a next-generation version of the DDI Lifecycle model. The DDI Core work reflects a change in this orientation, largely driven by the needs of the implementations which were used as the basis for the work. Given the Agile approach used, it was to be expected that some of the more critical aspects of the work would become evident as the test cases were examined.

What emerged from the development was not a replacement for the existing DDI specifications, but a model which can be used to extend the capabilities of systems which might already implement them. In the majority of the identified cases, some form of DDI was already used in the systems concerned (BLS time series was the exception here – although BLS uses DDI, it was not a large part of the model used for the DDI-CDI work). This included both DDI Codebook and DDI Lifecycle models.

The following section describes the major features of the DDI-CDI model.

A. Data Description and the Datum-Oriented Approach

The DDI-CDI model uses a datum-oriented approach to describing four major structural types of data, along with some of their important sub-types. It should be emphasized that this same approach could be applied to a larger range of structural types, and that during the course of the work various "hybrid" types were identified. The model in this sense provides a toolkit for describing data. As mentioned, the selection of types included in the model were driven by the specific test cases being examined, and are deemed to be those of most immediate utility.

The four major data structures are:

- **Wide Data:** Also known as "rectangular" or "unit-record" data, this is the data structure familiar from other DDI specifications. In this form, each row in a table represents a case, and each column a variable.

- **Long Data:** This data format describes a range of real-world types: event/spell data, streaming sensor data, etc. In this form, each observation is accompanied by fields which serve to identify it, and to indicate which variable it corresponds to. The data tends to form a very "long" table with a relatively small number of columns. This form is not fully describable in earlier DDI specifications.

- **Multi-Dimensional Data:** While DDI specifications have always been able to describe the tabulation of microdata into aggregates, there are other applications for multi-dimensional structures. Based on other common models for data of these types (e.g., SDMX, the DataCube Vocabulary, etc.) this model allows for a multi-dimensional description of data independent of any other structuring/processing of the relevant data. One important sub-type which is supported is time-series data.

- **No SQL/Big Data:** Data is sometimes stored and managed in relatively unstructured forms, and many popular tools today embrace this approach – it lends itself well to handling data sourced from social media and other, similar sources. In this structure, there are a range of schemes for identifying individual observations/values, but few intermediate structures. Data is notionally a "pool" from which selections of observations are made.

The datum-oriented approach allows all of these data structures to be described by recognizing that individual datums can play different roles in different structures. Any given value can function as a measure, a descriptor, or an identifier (etc.). By maintaining the identity of the datum across its use in different structures, and the role it plays in each, it is possible to fully describe data used for different purposes, and to document exactly how such transformations take place.

The datum-oriented approach represents a major expansion of the capabilities of the DDI model to describe the new forms of data which are becoming significant for research, both within the Social, Behavioral, and Economic (SBE) sciences and in other domains. It should be noted that DDI-CDI is "domain-neutral," applying equally well to data of these types from any discipline or domain as a result of its focus on describing data structure.

This capability is significant in data integration scenarios: while the data transformations between structures have always been possible, documentation of the roles played by their constituent values has not. The growing need for data integration both within and across domain boundaries in modern research requires this capability.

Once described using the DDI-CDI model, it becomes possible to programmatically map a given data structure into a different one using the same model (DDI-CDI) as a basis. Traditionally, such transformations have required labor-intensive manual work to understand the correspondences between structures; often, sufficient metadata has been lacking. DDI-CDI corrects this situation by supporting the formal description of these relationships between disparate data structures at an atomic level.

B. Process Description

Given the utility of the datum-oriented approach for describing data transformations, it is natural that the processes by which such transformation happens should also be documented. While DDI specifications have always supported the description of specific processes applied to data (cleaning, derivation, etc.) these have tended to be granular descriptions of specific programmatic functions, such as those described in a script used in a statistical package (etc.). DDI-CDI focuses instead on describing the higher-level processes which provide an account of the provenance of data. This often takes the form of the overall structure of the process which employs several lower-level processes in a sequence, etc. This level can be understood as the "business process".

There are two popular standard models for this type of process description: the Business Process Modelling and Notation (BPMN) standard, and the W3C PROV ontology. DDI-CDI uses these as the basis for its description of process, allowing for the integration of lower-level processes (whether in a proprietary language or using a platform-neutral language such as SDTL or VTL) in a fashion similar to that found in DDI Lifecycle.

A further requirement emerged during the MRT work: while many systems use the familiar style of procedural process description, in which a series of process steps are undertaken, controlled by flow logic, there is another style of process description which has emerged: declarative process description. Declarative process engines do not follow pre-determined flows, but take a set of criteria for completion, and apply a set of functions (a playbook) to them on an as-needed basis. This approach often relies on the use of parallel processing capabilities, turning the process engine into a "black box".

DDI-CDI does not attempt to describe the inner working of such engines, but does support the identification of the inputs, outputs, criteria, and playbook functions when documenting these types of processes. Further, it supports the description of "hybrid" processes which combine procedural and declarative approaches.

C. Foundational Metadata

The DDI 4 model produced a number of mature models for describing basic metadata constructs (e.g., classifications, concepts, variables, code lists, etc.) These were not the focus of the MRT effort but were applied to the description of data and process as outlined above.

Perhaps the most significant aspect of the DDI 4 model being applied to these use cases is the datum-oriented model itself. This provides an atomic view of data values independent of their structural context. During the course of the work, MRT refined this model, but it is fundamentally the same design which emerged from the earlier DDI 4 work.

Another major feature of the foundational metadata description is in the "variable cascade." Variables perform many different roles in different places, and the variable cascade provides a description of these different roles. This allows for variables – an important construct, but not a fully atomic one – to be modeled as appropriate for each different use. Relationships across waves of an ongoing study can be tracked; the difference between a conceptual variable and one encountered when exploring an existing data set can be understood, etc.

This model of variables is shared with DDI Lifecycle, having been incorporated in revisions to that specification. MRT used what was provided here by the earlier DDI 4 work.

D. Modelling and Technology Aspects

As was intended throughout the DDI 4 effort, the DDI-CDI model is a model-driven specification. Today, it is difficult to anticipate which technology platforms will be used to implement data systems. While XML remains the dominant paradigm for many traditional data management systems, many other types of applications use other languages and syntaxes. Because DDI-CDI uses a UML formalization, the determination of how the model is implemented – the syntax used – is left in the hands of the implementer. (It should be noted that many development packages can directly use a model expressed in UML as a basis for producing applications on a specific platform.)

DDI-CDI is published in the form of Canonical XMI – a standard for exchanging UML models, expressed in XML. This assures that it will work across platforms, a feature of the model which was tested by MRT. Further, the feature set of UML employed has been tested and documented, to further guarantee that it is useable across as many different UML tools as possible, whether these are modeling or development platforms.

An XML serialization has also been provided for DDI-CDI, using the binding identified in earlier DDI 4 work. W3C XML schemas and examples of their use are included in the current review draft of the specification.

While it was intended that an RDF syntax representation also be included, investigation here did not identify a solid approach, and resources were lacking to pursue this representation. This work remains to be done.

E. Alignment with Other Standards and Models

DDI has always taken other standards and specifications into consideration when developing its work products, and this has not changed with DDI-CDI. However, having a UML formalization of the model allows for a more direct alignment with many other specifications. The current DDI-CDI draft outlines the various forms which alignment with external models takes. In some cases this includes the direct incorporation of external models into DDI-CDI, as we see with the Process description and PROV/BPMN. In other cases, different approaches have been used. These are described in the draft specification.

In general, however, it is anticipated that DDI-CDI will need to interact with a large number of external models, including DCAT, Schema.org, various of the data-related W3C vocabularies (including DataCube, but also others such as SOSA/SSN, CSV on the Web, etc.), SDMX, JSON-based standards around data, PROV, BPMN, SDTL, VTL, and others. The intention of the model is that it support easy integration with external models and standards, as appropriate, and that this feature of the model be “future-proofed” to the greatest extent possible through an extensible basic design.

It should be noted that domain ontologies, classifications, thesauri, and similar constructs which detail semantics within a subject area are treated in DDI-CDI as they are in other DDI work products: it is assumed that users will employ whatever semantics are needed. The core model remains agnostic to these but provides for their use in describing data as appropriate.

VI. Relationship to Other DDI Specifications

The DDI-CDI development involved testing with applications which in most cases already used one or more other DDI specifications. What emerged from this was a DDI-CDI design which is aligned with these

standards where appropriate but is not duplicative of them. In essence, DDI-CDI is a complementary standard which supports the description of new forms of data, and of a new type of provenance/process description, but which does not replace the functionality provided by DDI Codebook or DDI Lifecycle.

Envisioned implementations include the addition of new types of data into management systems on the basis of descriptions using the DDI-CDI model, where the existing system may already use DDI Codebook or DDI Lifecycle, and the transformation into DDI-CDI of existing DDI Lifecycle/Codebook metadata for the purpose of integrating data with other data that might only be described using the DDI-CDI model. This last case would apply where non-traditional data sources (big data, sensors, etc.) were being integrated with more traditional ones.

Further, the process/provenance description capabilities of DDI-CDI are useful regardless of how the data itself is described: again, these descriptions can be used to complement those which are made using DDI Lifecycle or DDI Codebook. The alignment among the DDI-CDI standards in the area of data description provides the ability to transform one to the other where needed – it does not indicate that they are designed to perform the same function. Rather, they are intended to be used in a complementary fashion.

VII. Coordination with Other DDI and External Committees

A. Applicability of the Specification

DDI-CDI is different from other DDI work products in that it is more domain-agnostic than earlier specifications. While the application of DDI Codebook and DDI Lifecycle has been broad, encompassing the SBE sciences, health research, and official statistics, DDI-CDI has an application which is potentially even broader.

This was not initially seen as a requirement, but the incorporation of data from non-traditional domains into social science research demands the ability to describe that data. Once you are able to describe external data, the model is of necessity applicable outside of the social sciences as well as within it.

Thus. the model supports the integration of data across domain boundaries. This fact has led to new challenges and opportunities both. For the DDI Alliance, interest from outside the traditional user community needs to be managed, but with this comes the potential for a standard which is used more widely and reaches a broader audience.

DDI-CDI has been written in a fashion which will hopefully make it more approachable to those both within the traditional DDI user community, and to those outside it. Emphasis has been placed on providing more examples within the specification, and at different levels of complexity.

B. Coordination with DDI Alliance Committees

The MRT group has met with and continues to engage with the Technical Committee, the Marketing Committee, and the Training Committee of the DDI Alliance, and there is significant cross-membership among these groups. It is hoped that efforts to solicit technical review from groups outside the traditional DDI user community will then become opportunities for training and marketing on the part of the Alliance. At this point, this seems likely.

C. Coordination with External Groups

External interest in DDI-CDI has primarily come from two sources: the cross-domain workshops held at Schloss Dagstuhl in 2018 and 2019, and from the soon-to-be-launched Decadal Programme of CODATA, which sees in DDI-CDI a tool for supporting its primary goal of cross-domain data sharing at a global level. (CODATA is the data arm of the International Science Council, spanning social and hard sciences both, and working closely with groups such as GO FAIR and RDA.)

Already, CODATA has offered to help recruit interviewers from outside the traditional DDI community for the current public review, and is further discussing how the various training efforts it supports in the area of Research Data Management could benefit from collaboration with the DDI Training Committee. These discussions are currently ongoing, but it is hoped that the visibility and use of DDI will benefit from having a domain-agnostic specification which is well-suited to describing the integration of disparate forms of data.

VIII. Current Status and Future Plans

Currently, the DDI-CDI specification is available for public review, which will continue through the end of July 2020. During this period, a number of introductory webinars will be offered to help potential reviewers engage with the specification and to comment meaningfully on it.

These webinars are partly aimed at potential users outside the DDI community who may not know of the DDI Alliance or its specifications, but who may be interested in it. Their input is critical, as they may represent domains whose data must be supported by DDI-CDI in order for it to be integrated with more traditional SBE data.

Once the review is complete, a revised version of the specification will be produced, and a further round of focused review conducted by reviewers who have been recruited. Initially, this second review round was to be conducted at a week-long intensive workshop at Schloss Dagstuhl, but whether this will be possible in light of the COVID-19 pandemic is doubtful. Regardless, an intensive review is planned either as a face-to-face workshop, as a virtual activity, or as a combination of the two.

Following this, any further needed changes would be incorporated, and the specification prepared for final release. Anticipated release would occur in the very start of 2021, given time for packaging and review following the second round of review in late 2020.

It is hoped that early adopters will begin implementation of the specification before the production release is complete, to help guarantee that it is of the desired quality.

IX. Summary

DDI-CDI provides a capability for describing a broad range of data types which have not been supported by earlier DDI specifications. It provides a capability to describe process in support of data provenance. Further, it leverages the work of the DDI 4 effort in being model-driven, and builds on some of the strengths of that model such as the variable cascade. The datum-oriented approach makes it a suitable tool for describing data integration both within domains and across domain boundaries.

These are important features, and they expand the utility of the DDI product suite in a significant fashion. It is hoped that, by being domain-agnostic, DDI-CDI can also support applications within the SBE

sciences which need to incorporate data from external domains, and to provide the same capability in reverse, to those external domains which use SBE data.

DDI-CDI does represent an application of the core DDI 4 model in a practical, useful way, as was originally intended. What has also emerged from the development process is a specification which complements the existing XML standards and can be used in combination with them by those organizations which have already made an investment in DDI.

DDI – Cross Domain Integration: Introduction

Contents

I. Overview

The DDI – Cross Domain Integration (DDI – CDI) specification provides a model for working with a wide variety of research data across many scientific and policy domains. It provides a level of detail which supports machine-actionable processing of data, both within and between systems, and is designed to be easily aligned with other standards.

It focuses on the key elements of the data management challenges facing research today: an exact understanding of data in a wide variety of formats, coming from many different sources. Two elements are critical for dealing with these challenges: a flexible means of describing data that can reveal the connections between the same data existing in different formats, and a means of describing the provenance of the data at a detailed (but comprehensible) level: the processes which produced it must be made transparent.

DDI - CDI covers these areas in a fashion intended to make it optimally useful to modern systems, which often employ a variety of models, and comply with a range of related specifications for both functions related to data description and process/provenance. The model is designed to be easy to fit into such systems, by aligning with relevant external standards, and to be alignable with them into the future.

II. Contents of the Specification

The specification is separated into several documents and files, appropriate to the material covered. These are described in the document “DDI-CDI Overview PR 1.pdf.”

III. Purpose

The DDI - CDI specification describes a model and supporting elements for implementing it in the areas of data description and process/provenance. It is not intended to supplant existing specifications for these purposes, but to fill in the information which such specifications often do not capture. For data, this is the description of a single data point – a datum – which can be used to play different roles in different data structures and formats. For provenance and process, this is the packaging of specific machine-level processes, which may be described in many different ways, into a structure which relates them to the business processes described at a level understandable to human users.

In order to serve this purpose, the DDI - CDI specification uses a Unified Modeling Language (UML) formalization so that it can be mapped against other models within systems more easily. Several different syntax expressions of the model are made available to support implementation.

Several important features of the specification can be highlighted, to show how it serves this purpose:

- Domain-independence
- Datum-Oriented Data Description
- Provenance and Process Description
- Foundational Metadata
- Interoperability, Sustainability, and Alignment with Other Standards

Each of these will be addressed in more detail, and an outline of the specification documents is presented.

IV. Key Features of the Specification

A. Domain Independence

DDI - CDI is designed to be used with research data from any domain. In order to do this, it is fundamentally based on the structure and other generic aspects of the things it describes. It does not attempt to be a domain model of semantics, nor a model specific to the life cycle of a particular domain of science or research. [Historically, DDI has focused on the Social, behavioral, and Economic (SBE) sciences some types of health research – to see how DDI - CDI relates to other DDI specifications, see the last section in this document.)

DDI - CDI is intended to be complimentary to (and used in combination with) other standards and models which focus more on domain-specific aspects (such as semantics and life-cycle models). Such generic elements such as classifications and variables are given a detailed formal treatment but are agnostic as to semantics and concepts. It is left to the user to employ whatever semantics and concepts are demanded by the data with which they are working.

This feature of the specification makes it well-suited to combining data coming from more than one domain or system, to allow a description of it that supports systems which perform data integration, harmonization, and similar functions. Cross-domain data sharing is becoming increasingly common, and DDI - CDI is intended to provide support for this type of application.

B. Datum-Oriented Data Description

DDI - CDI embraces a form of data description which is based on its atomic components: individual datums. Any given datum can play different roles in different formatting of the same data set, depending on how it is processed and transformed. In order to retain the continuity of a given datum across different formats and throughout a series of processes, DDI - CDI allows it to be described playing different roles in different structures.

DDI - CDI provides four basic types of structural description for data sets: wide data, long data, dimensional data, and key-value data. These four types (and their sub-types) provide coverage for many common data formats today. While not comprehensive, they cover the majority of cases that the developers of this specification have seen. These include many of the newer forms of data such as streaming data, "big" data, registers, and instrument data. The underlying approach is one which could – and may be – expanded in future. By assigning appropriate roles to datum in each of these different formats, however, it is possible to understand how data passes from one form to another.

C. Provenance and Process Description

If we are to fully understand data, we also need to know how it has been processed and transformed. Given our ability to describe how a different datum can be used in different data sets, it becomes desirable to understand also how those data sets relate to one another in terms of the processes which use them. This can be understood as an important aspect of data provenance.

There are many different ways of describing process and provenance. Popular models include the Business Process Modelling and Notation (BPMN) standard and the PROV Ontology (from W3C). There are a multitude of syntaxes for driving data transformation, cleaning, and analysis in packages such as R, SAS, Stata, MATLab, SPSS, Python, and so on. There are also some emerging standard models for specifically describing such specific processes (eg, SDTL, VTL).

DDI - CDI attempts to do something which compliments the use of such models, by connecting specific processes interpretable by machines at the lowest level (described in a package-specific syntax or language) with the higher-level flows which combine these into human-readable documentation of business processes. Both traditional linear processing and the newer declarative processing approaches are supported.

D. Foundational Metadata

In order to formally describe data at a detailed level, there are many component elements which themselves must be modelled. Statistical concepts and their various uses – including as categories and variables – are a core part of this, but the range is broad. These components are included in DDI - CDI as "foundational metadata."

Terminology for such constructs varies widely across domains. DDI - CDI has attempted to provide common terms for these components, and to adopt common models from other standards where it seemed useful.

One area which deserves particular attention is the "variable cascade" – a model for how the different types of variables relate to each other, and how they reflect the way data is described at different points

in its creation, processing, and use. While many different models have a "variable" of some form, the one presented in DDI - CDI reflects the experience of working with this important construct in many of the specifications and standards which have preceded it. It is a nuanced view of how variables relate and are are understood across different systems, and – although not simple – it is a powerful model which helps solve some of the commonly encountered problems in data description and management.

E. Interoperability, Sustainability, and Alignment with Other Standards

DDI - CDI is fundamentally a model which is intended to be implemented across a wide variety of technology platforms, and in combination with many other standards. Models, and specifications. To support this use, it is formalized using a limited subset of the Unified Modelling Language (UML). The model is provided in the form of Canonical XMI – an interchange format for UML models supported by many different modelling and development tools. Further, a syntax representation is provided in XML, so that direct implementation of the model is possible if needed.

The platform-independence of the model makes it more easily applicable across a broad range of applications and helps ensure that it will be sustainable even as the technology landscape evolves.

DDI - CDI builds on many other standard models and is aligned with them where appropriate. This is shown in the model itself, where formalizations from other models and specifications are refined, extended, or directly used. The specification includes a description of what these other standards and models are, and how they are used in DDI - CDI.

V. DDI - CDI and the Suite of DDI Specifications

DDI - CDI is a different type of specification than its predecessors. It is not a continuation of or replacement for earlier DDI specifications such as DDI Codebook or DDI Lifecycle. It is intended to be complementary to these specifications for those applications – mostly in the SBE sciences – where DDI is used.

DDI – CDI builds on the work of many years in the DDI 4 project and brings some of the strengths of that effort to light. In this, it shares many features with later versions of DDI Lifecycle, which has also incorporated some of that work. Notably, the "variable cascade" comes from earlier DDI 4 models, as does the overall approach to describing non-rectangular data.

The DDI - CDI Model is the first specification produced by the DDI Alliance which uses a conceptual model expressed in UML as its basis. It is intended to describe many of the types of data which earlier DDI specifications describe. Due to the way in which data today is increasingly used across traditional domain boundaries, however, DDI - CDI is also (and of necessity) capable of describing data from many related domains.

The purpose of the specification differs somewhat from the earlier DDI Codebook and DDI Lifecycle specifications. Due to changes in the way in which information technology is applied to research and statistics, some new features are emphasized. Notably, the diversity of data types analyzed in a given project has increased, and the range of sources for that data has grown, with corresponding changes in the technology used to manage it.

The functional goal of the specification is also different: where DDI Codebook was an XML representation of a data dictionary, and DDI Lifecycle a more complex model designed to support metadata from data conception and capture through publication and reuse, DDI - CDI is an attempt to describe data and its provenance independent of these contexts.

Both DDI Codebook and DDI Lifecycle combine the description of structure (e.g. a table of records) and the description of meaning. In both, the primary structural form is a table or a cube. A variable and a column are basically synonymous. DDI - CDI disentangles structural description from description of meaning. This allows description of structural forms like tall tables or key-value stores.

The growing demand for data from different sources, and from external domains, requires that some different types of data be described. The provenance of this data – that is, the processes by which it has been assembled for use – are of increasing importance in understanding what it is and how it can be used. While traditional SBE data was often collected using questionnaires, alternate sources of data such as registers and sensors are becoming increasingly common and have in some cases always been typical. Completely new types of data from social media and other "mined" sources are also increasingly used.

The DDI - CDI model applies the important features of the pioneering (but unreleased) "DDI 4" work to these functions: describing various types of data in a way which makes them subject to integration and transformation into useable forms, and providing the information needed to understand their origins and provenance.

Because the way in which such a model can be implemented is more variable than it is for traditional SBE data management systems, the emphasis in DDI - CDI is on a model, formalized in UML, and made available using the Canonical XMI format which supports the exchange of UML models between various tools, including both modelling and development environments. While XML is still supported, it is no longer the canonical format for the specification.

DDI - CDI is aligned with earlier DDI specifications, most notably DDI Lifecycle, as it is anticipated that it might be used as an integration model for systems based on these earlier specifications. The intention is that DDI - CDI be a tool which can supplement systems using earlier versions of DDI, enabling them to better handle new types of data.

DDI-Cross Domain Integration: Detailed Model

Contents

I. The Upper Model

The purpose of this section is to show how DDI – Cross Domain Integration (DDI – CDI) fits into the overall context of research and the metadata which is used to describe research activities and resources. The upper model touches on some areas which are not covered in detail by other parts of DDI - CDI, but which may be important in terms of understanding how it can be implemented in systems or with other standards that do cover those areas. The Upper Model is merely diagrammatic: it is not included as part of the detailed DDI - CDI Model and is not found in the syntax representations. It is intended to provide context for how the model may be used in relation to other real-world entities within systems.

Note that an "upper model" (also known as an "upper ontology" and "foundation ontology") functions "to support broad semantic interoperability among a large number of domain-specific ontologies by providing a common starting point for the formulation of definitions. Terms in the domain ontology are ranked under the terms in the upper ontology, e.g., the upper ontology classes are superclasses or supersets of all the classes in the domain ontologies" (Wikipedia).

A Research Program for the purposes of this model is an undertaking intended to produce and/or analyze data which measures real-world phenomena for the purpose of answering questions about policy and science, understood in its broadest sense. It is the context for data capture, data management, data analysis and /or data dissemination, depending on the actual scope of the Research Program.

The diagram below shows the Research Program together with a Research Methodology that guides it:

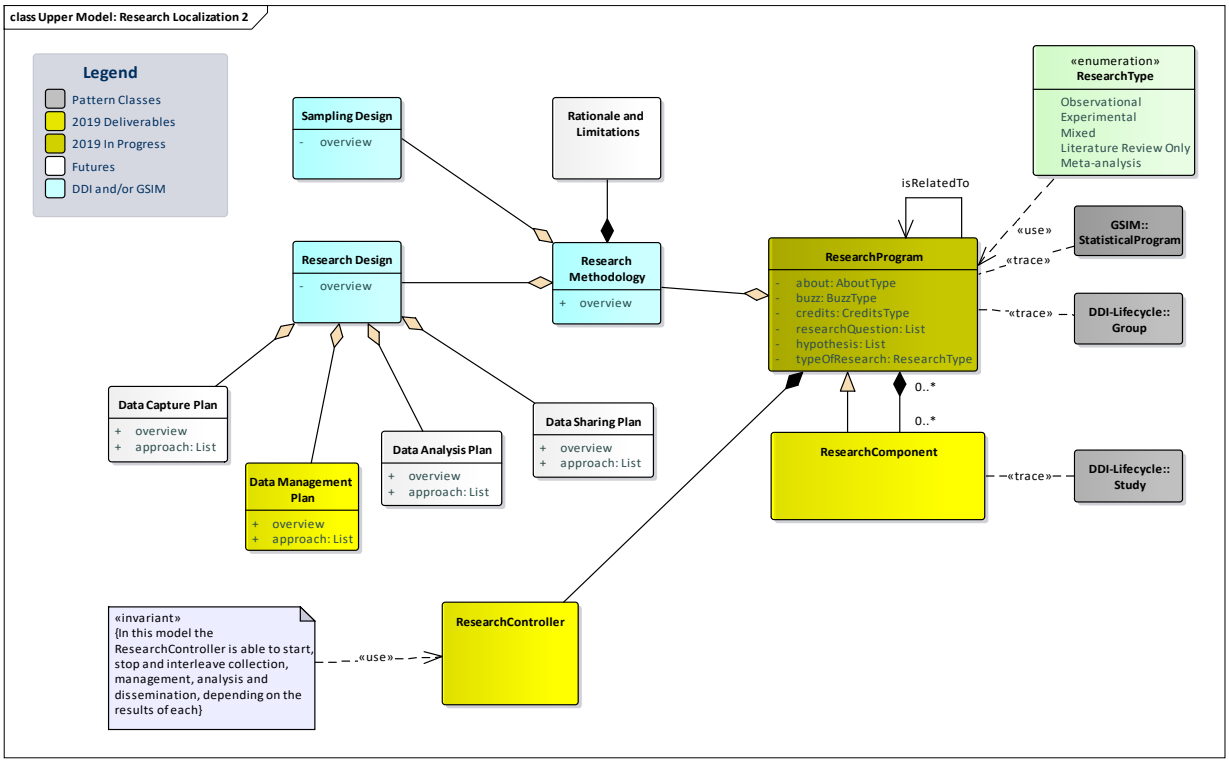


Figure 1: Research Program and Research Methodology

In this model the Research Program corresponds to a GSIM Statistical Program and the Group from DDI Lifecycle. Here a Research Component might be a "round" in a longitudinal study. However, a Research Program need not be a longitudinal study. Instead of a time series, it could also be the umbrella for a group of more or less related snapshots -- a research "mosaic", if you will -- that an organization undertakes, depending upon what the Research Program and its Research Components are "about".

The "about" property group of a Research Program and its Research Components is a structured set of data based on terms from DDI Codebook, Dublin Core and Schema.org, assuring comparability across many standards. Comparability is assured both within the DDI family of standards as well as between DDI and other standards. Note that the "about" of a Research Program and its Research Components includes "coverage" and "provenance" at various levels of specificity in line with Dublin Core, DDI Lifecycle, and Schema.org.

A Research Program also has "buzz" and "credits" property groups. "Buzz" comes from Schema.org and is the profile that a Research Program cuts in social media – its interaction statistics, audience characteristics, comment characteristics, trolls and similar aspects. Between "about", "buzz" and "credits" the Research Program supports the construction of many types of bibliographic citations. (To understand how these property groups were developed, see the document "DDI Cross Domain Integration: Architecture and Alignment with Other Standards" in this review package.)

A Research Program is informed by and conducted according to a Research Methodology, which involves the plan for how it will be conducted, represented by the Research Design class. Research Methodology here is generic. It is patterned after the Research Methodology section or chapter in a thesis or a research paper in line with the recommendations of many scientific and governmental organizations. It is not intended to be machine actionable. Indeed, DDI - CDI does not provide a detailed model of methodology or data management planning. Instead, it focuses on the data and processes which such methodologies and plans might involve. One exception to this overall approach is that Sampling Design is broken out. Sampling Design is a placeholder for a sampling plan model able to describe sampling plans for surveys and other statistical activities for all kinds of samples including probability, non-probability and multi-stage / multi-frame.

A Research Program is controlled by a Research Controller – in PROV-O terms an Agent. The Research Controller starts, stops and interleaves data collection, data management, data analysis and data dissemination, depending on the results of each, in line with the Research Methodology.

The diagram below shows in more detail how the Research Controller relates to other points of focus within DDI - CDI, notably process and data description.

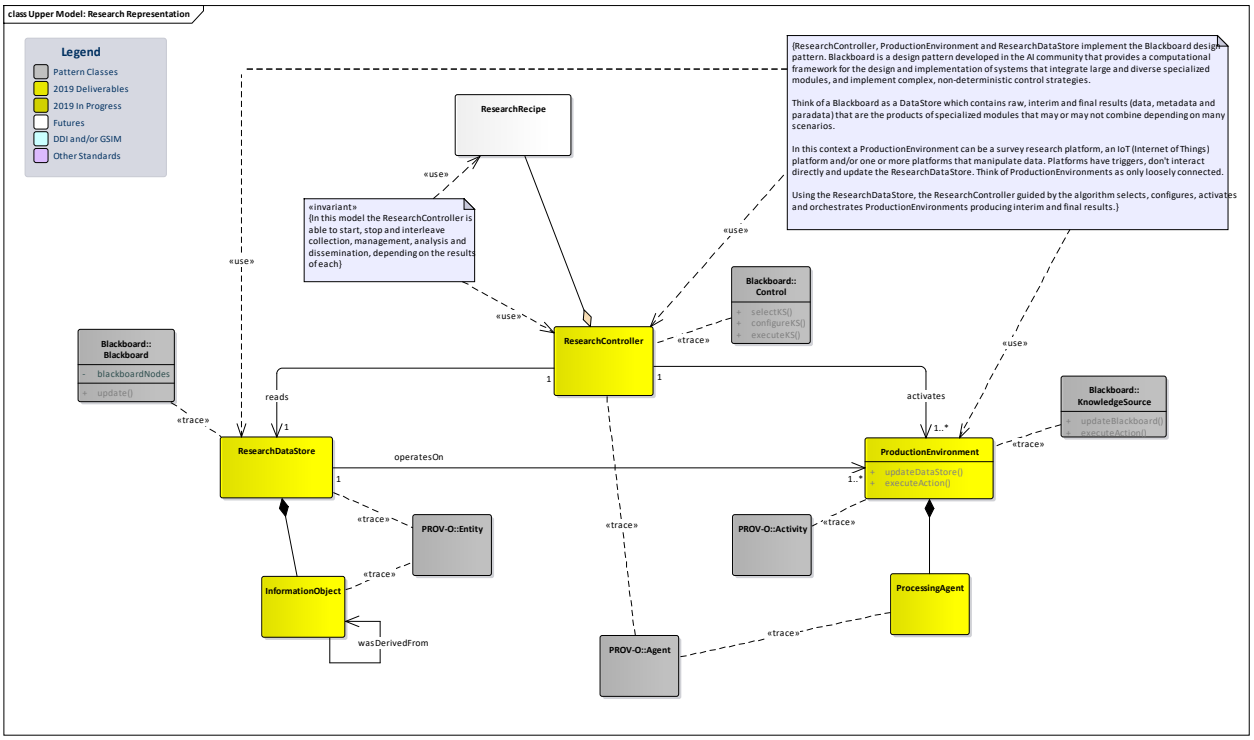


Figure 2: The Research Controller

The Research Controller interacts with two important parts of the model -- the Research Data Store and the Production Environment. The Research Data Store may be a repository, a virtualized data store or a metadata store that represents the data at its many stages of development which are used by and produced during the Research Program. DDI - CDI provides a model for describing many of the types of data which Research Data Stores contain. This model is extensive and supports the integration and transformation of the different types of data required.

The Research Controller also interacts with the Production Environment. DDI - CDI includes two types of production environments for now – the Data Capture Platform and the Data Processing Platform.

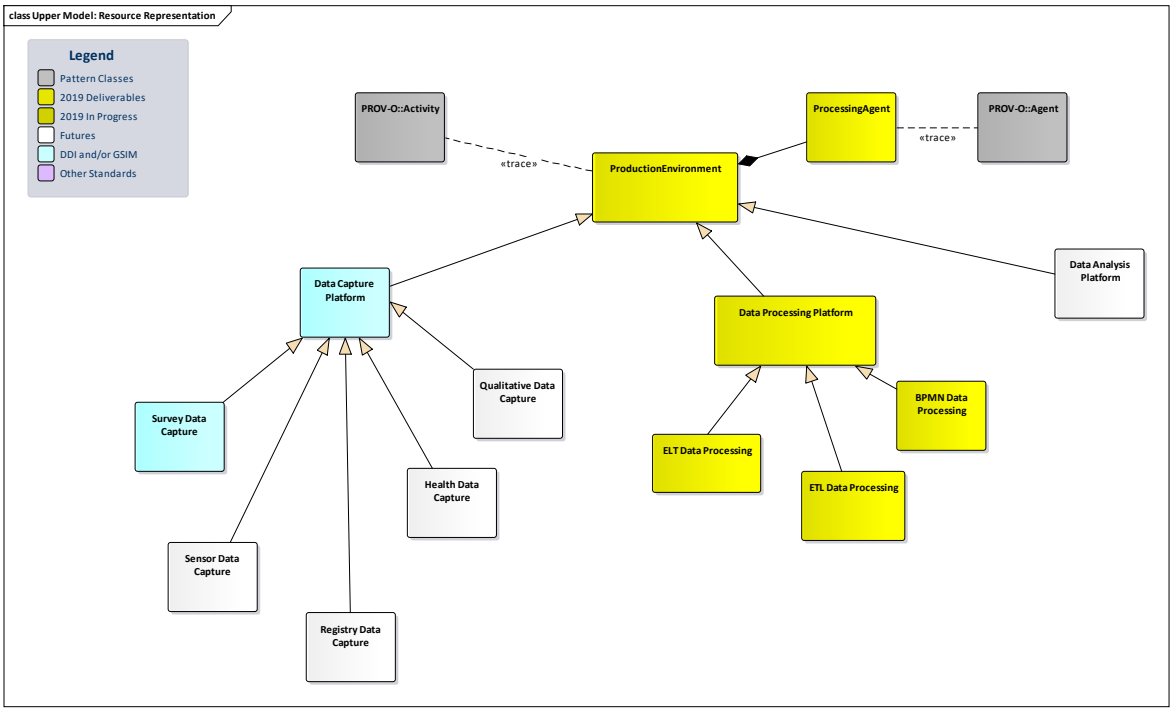
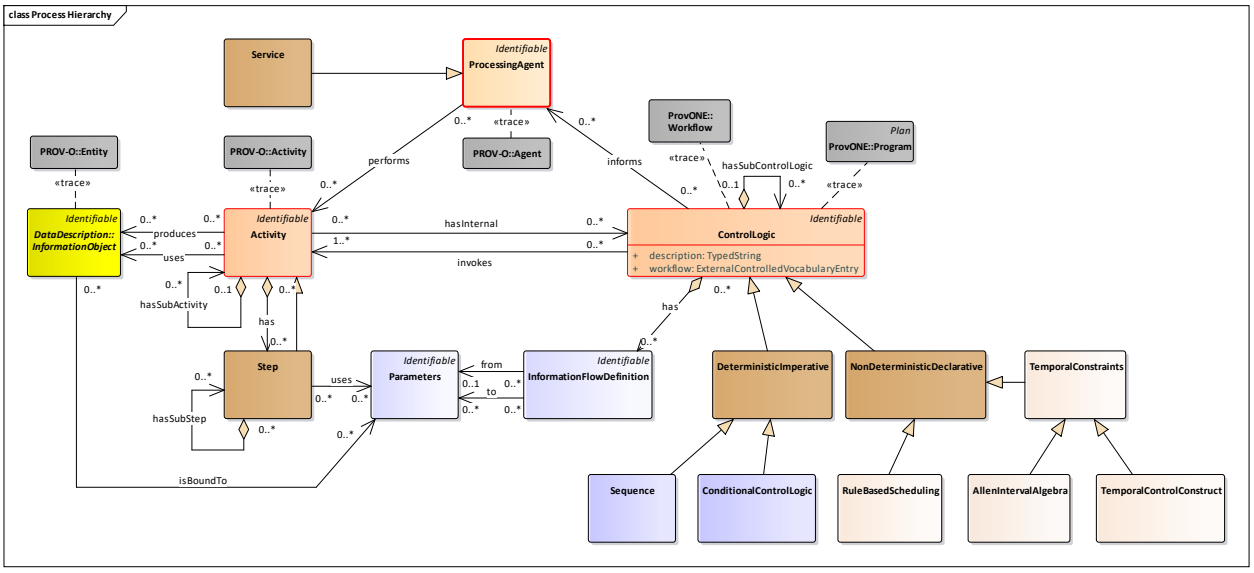


Figure 3: Production Environments

The Data Capture Platform includes Survey Data Capture together with other data capture types. Several types of Data Processing Platforms are depicted here including the ETL (Extract, Transform, and Load) , ELT (Extract, Load, and Transform), and BPMN Platforms. These types and the several data capture platform types shown here are placeholders for models that may have been developed by other standards. Treatment of these placeholders by the production system and at instantiation time by tools developed in various communities of practice is discussed in the Architecture document sections on standards alignment.

What DDI - CDI, however, does include is a detailed Process Model. What is presented here is a very high-level description. The detailed description can be found in Section IV of this document.



F*igure 4: Process Model High Level*

The DDI – CDI Process Model is intended to be usable in every type of Production Environment. It supports the description of both deterministic and non-deterministic control systems. Deterministic control systems use deterministic control logic. Non-deterministic control systems include rules-based systems and systems that employ machine learning.

Note that the DDI Core Process Model traces to PROV-O and an extension of PROV-O called ProvONE. This is because the process descriptions in the DDI - CDI model are intended to support the description of process as provenance documentation. This is significant – in past versions of DDI, the process model has been designed to both describe workflows (DDI 4 Prototype) and the flow of questionnaires and related processing (DDI Lifecycle, DDI 4 Prototype) in sufficient detail that they can drive the execution of data processing workflows and questionnaires. In DDI - CDI the focus of the process description is restricted to just the documentation of provenance. Note that DDI Core, like ProvONE, extends the "entity" in PROV-O to data at the variable level, the documentation of specific workflow types and the evolution of workflows as occurs in machine learning.

The Research Data Store comprises a set of Information Objects of different types. Significant among these are data sets. DDI - CDI provides a detailed description of data and the structures which are used by different types of data sets.

Some of the most significant Information Objects – those related to describing data – are a major focus if the DDI - CDI model. The diagram below shows a more detailed view of how data fits into the Production System.

Note that DDI - CDI Information Objects relate to classes in some other popular models: PROV-O provides us with the Entity, to which Information Objects correspond; they can also be seen as equivalent to GSIM Information Resources.

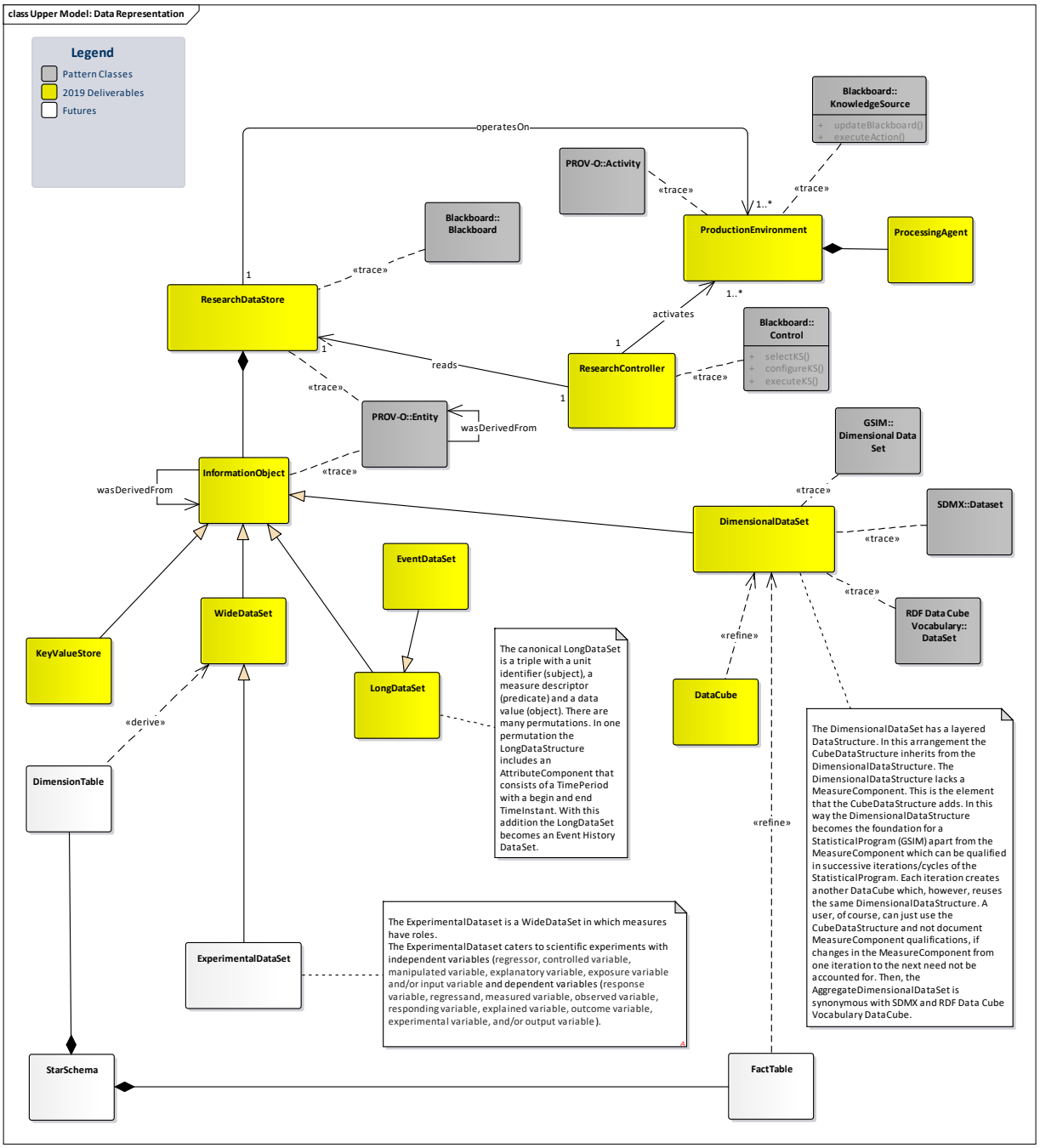


Figure 5: Data Representation

The mechanism for describing data structures in the DDI - CDI model provides four basic types: Wide Data, Long Data, Multi-Dimensional Data, and Key-Value Data. These types represent different styles of describing data structures, using a consistent set of components for identification, grouping observations into records, adding descriptive fields, and so on. The differences between each type of data description are found in the roles played by these different components.

The four types are characterized as:

Wide Data: This is a way of describing traditional rectangular unit-record data sets. Each record has a set of observations about a single unit. The record has a unit identifier and a set of measures and/or descriptors which are the same for each unit. The unit identifier can be used as an identifier for the record, because each unit has only one.

Long Data: This is a technique for describing many common types of data, including sensor data, event data, and spell data. In this form, each record has a unit identifier and a set of measures and/or descriptors, but there may be multiple records for any given unit. The identification of the record is a combination of the unit identifier and one or more other fields.

There are two refinements of the Long Data class which further constrain it to correspond with Event Data (defined as Long Data for which an observation time is provided as a single point) , and Spell Data (Long Data with fields for start and end times bounding a period). In both of these, the record identification involves time as well as the unit identifier.

Multidimensional Data: Multi-dimensional data is data in which observations are identified using a set of dimensions. These values both identify the cell and serve to describe the measured population. Records may be organized in various ways and may include descriptors as well as their dimensions and measures. It is common to view such data sets as multi-dimensional Cubes, and also to describe them as Time Series. These specific approaches are defined as sub-types of DDI - CDI's multi-dimensional data.

Key-Value Data: Key-Value Data is data which consists of a set of measures, each of which is paired with an identifier. Descriptors may also be attached to these pairs. Such data is often organized in more complex ways when it is used but may be stored or exchanged using this simple construction.

II. Foundational Metadata: Concept, Datum, and Variable[1]

A. Introduction

This section explains concepts, data, and variables as used and described in DDI - CDI. It is detailed and technical, but the language and ideas are accessible. They are based on everyday experience; however, the approach may be unfamiliar. The result provides a thorough understanding of concepts, data, and variables as used in DDI - CDI. The approach uses the theory of terminology as described in ISO 704 – *Terminology – Principles and methods*.

The underlying theory for understanding data and variables is the same as that for concepts and terms. We develop this connection. We start with a full description of concepts, what they are, how they are structured, and how people refer to them. From this we show that data are a kind of terminology, and the connection between concepts and their corresponding objects is precisely what variables do.

Data are often described by what they do, the operations and statistics available to process them. The terminological approach is an attempt to define what data are.

Variables are described in DDI - CDI through levels of specificity. This is known as the variable cascade, and it enhances reuse of metadata, an important principle of metadata management. How the cascade ties back into the terminological view of data and variables is described.

[1] Includes material taken from an unpublished research paper by Frank Farance and Dan Gillman.

B. Theory of Terminology

The theory of terminology for special language is described in ISO 704 – *Terminology – Principles and methods*. This section is a reformulation of that standard to data and variables. The ideas of concept and object are used throughout.

Readers might find this overall section very philosophical. It is not intended to be that way. We adopt a mentalist position for concepts and nothing more. This corresponds to experience. Likewise, objects are very generally defined, and they correspond to things in the world that people and systems use. We hope this approach allows the reader to maintain an intuitive understanding.

1. Objects

An object is anything perceivable or conceivable. This is a very general definition, and it implies that the idea of object, for our purposes, is the most general thing we consider. Each thing, physical or not, is an object.

Perceivable objects are those detectable through one of the five human senses. Any physical object in the world is perceivable, mostly through sight and touch, but the other senses may be used as well. For instance, a sound is perceivable through hearing. An object may also be perceived through some detector. Examples include voltage and current (in electricity), and they are perceived through instruments.

Objects may also be conceivable, and these come in two main kinds: abstract and imaginary. Examples of abstract conceivable objects are variables, laws, and numbers. Examples of imaginary conceivable objects are unicorns and hallucinations.

2. Properties and Characteristics

A **property**[2] is a determinant, the result of a determination either directly or indirectly about some object. Note, this term is used in many other subject areas, and its meaning here is close to the others. However, there is a precise and specific meaning being used here.

One form of determination is through observation – something humans perceive through their senses. Noticing the color of a person's eyes is an observation or direct determination of the eye color of that person. Another form of determination is through detection by an instrument. An oral thermometer is an instrument that detects internal body temperature of a person. Observing a reading on the thermometer is an indirect determination about the internal temperature of a person. The specific observed eye color and internal body temperature are properties of a person.

It is through properties that enable us to describe and make distinctions between objects. For instance, one person may be 185 cm tall, have brown colored eyes and hair, and have medium brown colored skin. Another may be 170 cm tall, have blue colored eyes and blond hair, and have very light brown colored skin. These properties of each person serve to help distinguish between the two.

[2] The term property is not defined in ISO 704.

Since the examples above use perceivable objects, it is important to note that conceivable objects have properties, too. For instance, consider the rational numbers “three and fourteen hundredths” and “negative seventeen”. In the same way as with perceivable objects, properties of conceivable objects are the results of determinations about these objects. Here, the “sign” (in the mathematical sense) of the numbers is a property of them. The “sign” of 3.14 is positive, and the “sign” of -17 is negative.

A **characteristic** is a determinable. A determinable is something capable of being determined, definitely ascertained, or decided upon. It implies a question. Eye color, for instance, is a determinable, and applied to a person begs the question of what the color of the eyes for that person is. It is capable of being ascertained by looking into a person’s eyes to determine their color. A property, on the other hand, is the outcome, or determinant, and it is the answer to the question posed by the determinable. A determinant is an element that determines or identifies the nature of something. Blue is a determinant for eye color. So, a characteristic has the capacity for being determined (determinable), whereas the property is the result of a determination (determinant). Some characteristics of a person are height, eye color, hair color, and skin tone. Examples of corresponding properties, taken from the paragraph above, are: height has the properties 185 cm and 170 cm; eye color has the properties brown and blue; hair color has the properties brown and blond; and skin tone has the properties medium brown and very light brown.

A set of properties corresponds to a characteristic. These properties (those in a set) form an extensional definition (See sub-section on Definitions) of the characteristic they correspond to. In Examples 5 and 6, different sets of properties may correspond to the same characteristic, depending on needs. In addition, the same property may correspond to two characteristics. The following Example 1 illustrates this.

EXAMPLE 1: A property may correspond to two characteristics. Consider the following characteristics: height (of a person) and length of the diagonal (of a television screen). The property 60 inches (5 feet or 152.40 cm) corresponds to both characteristics. Some people are 60 inches tall and some large widescreen television sets measure 60 inches diagonally across the screen.

3. Concepts

A **concept** is a unit of thought differentiated by a set of characteristics. Consider the concept “person”. The characteristics of a person include being designed to stand upright on two legs, ability to talk, age, marital status, and skin tone. There are many others.

Some characteristics are indispensable for understanding a concept. These are the **essential characteristics**. A **delimiting characteristic** is a characteristic used to distinguish it from a generic concept. For example, an essential characteristic of people is they are designed to stand and walk upright. This is also a delimiting characteristic since it distinguishes people from other primates.
The **intension** of a concept is the set of characteristics associated with the concept. The **extension** of a concept is the totality of objects to which a concept corresponds.

A **defining characteristic** is a characteristic which is representative of objects in the extension of a concept. A defining characteristic of people is that they stand and walk upright. Not every person is capable of walking and standing upright, even though they are designed that way. Paralyzed or injured people may not be able to stand.

Characteristics and properties are concepts, and each kind plays a role. The role is how the ideas are distinguished. The role for a characteristic is determinable, and that for a property is determinant.

Example 2 illustrates the importance of establishing essential characteristics for a concept. In particular, the addition of a single characteristic may have profound influences on the objects in the extension of the concept. Adding or removing characteristics often affects the meaning of a given concept, changing the concept itself. Thus, the extension would be expected to change.

EXAMPLE 2:

The concept of planet was revised in 2006 by the International Astronomical Union. This revision resulted in the elimination of Pluto as one of the planets in the solar system. Pluto was long considered the ninth planet in the solar system, but some astronomers questioned this classification. Several properties Pluto possesses differ markedly from those of the other planets. Additionally, recent advances in astronomy - much better telescopes and vastly improved computation - showed there are many more celestial bodies that could be considered planets if Pluto remained one. Therefore, a concerted effort was made to define “planet” in a more limiting way.

The concept of a planet is now defined by these four essential characteristics: A planet is a celestial body that

1 Is in orbit around a star
2 Contains sufficient mass to maintain a nearly spherical shape due to its own gravity
3 Is not massive enough to cause thermonuclear fusion in its core
4 Has “cleared the neighborhood”, i.e., become gravitationally dominant, so the only other bodies in its vicinity are its satellites

This fourth characteristic is what eliminated Pluto.

A **general concept** is a concept which corresponds to an indeterminate number of objects which form a group by reason of common properties. An example is the concept “planets in our solar system”. An **individual concept** is a concept which corresponds to only one object. An example is the concept “Saturn”. In other words, a general concept may have any number of objects in its extension, and an individual concept must have exactly one object in its extension.

Note, a concept might be so defined that there exists only one object in its extension even though the possibility for more exists. This is still a general concept. For example, the notion “all planets with one moon” is a general concept. Even though there is one known planet with one moon – Earth – the possibility there are more cannot be ruled out.

The following Figure 6 shows the relationships between concepts and characteristics on the one hand and objects and properties on the other. The figure illustrates the correspondence between a concept and all the objects in its extension. The parallels between this construction and how data are obtained

through surveys, experiments, clinical settings, and other kinds of observations is clear. This parallel will be discussed in subsequent sections.

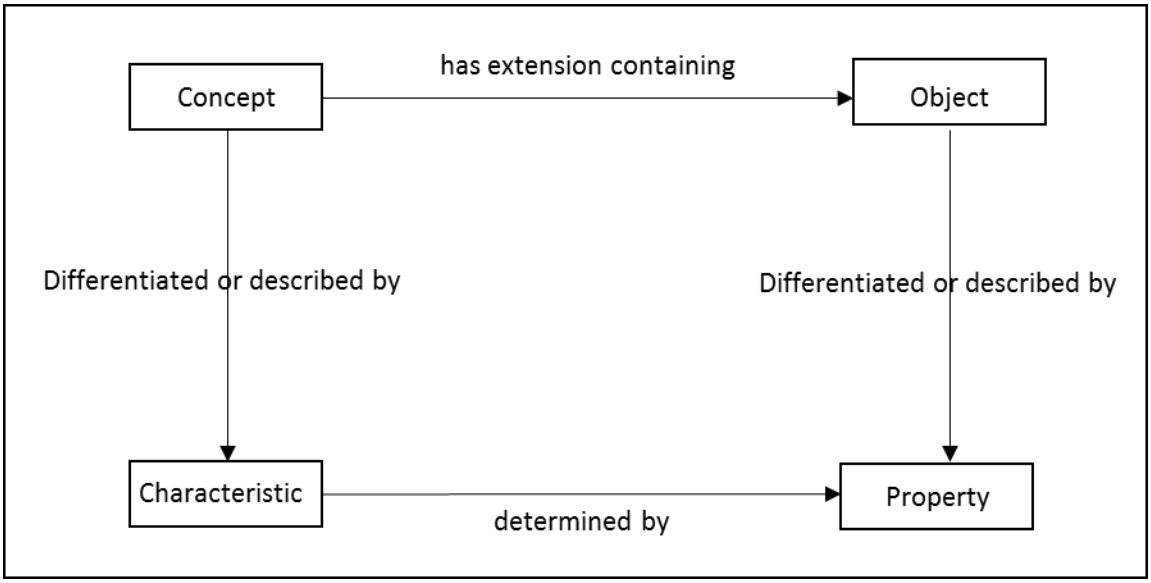


Figure 6: Concept - Object Correspondence

4. Signifier, Signified, and Sign[3]

A **signifier** is a concept whose extension is restricted to perceivable objects. An object in the extension of a signifier is a **token**. For instance, the objects **5** and **5** are both tokens of "the numeral five", a signifier. A signifier has the potential to refer to something. Typically, in statistical offices, the tokens of signifiers are alphanumeric strings.

A **signified** is an object intended to be denoted by a signifier. Any concept, which is also an object, may be a signified. A **sign** is the representation of a signified by a signifier, which denotes it.

See Figure 7 for a pictorial explanation of signs, consistent with the wording in this section.

[3] This is outside the scope of ISO 704.

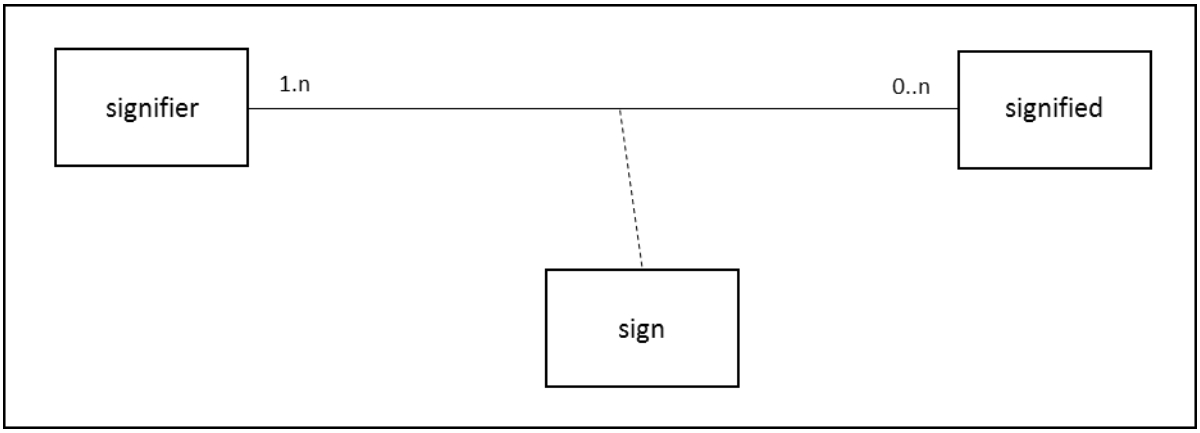


Figure 7: Structure of Signs

5. Kinds of Signs

The following is a list of kinds of signs for which the signified is an object:

- A **label** is a linguistic sign for an object
- A **name** is a non-linguistic sign for an object, where the signifier is an alphanumeric string
- An **identifier** is a label or name intended to be used for dereferencing
- A **locator** is an identifier with a known dereferencing mechanism

The following is a list of kinds of signs for which the signified is a concept:

- A **designation** is a sign for a concept
- A **code** is a non-linguistic designation for which the signifier is alphanumeric
- An **appellation** is a linguistic designation for an individual concept
- A **term** is a linguistic designation for a general concept
- A **numeral** designates a number, where a number is a concept, and the tokens for numerals are numeric strings

Terms, numerals, and codes are typically used to designate values (See section on Data).

6. Definitions

A **definition** is a descriptive statement which serves to convey the meaning for a concept, and it differentiates it from related concepts. There are 2 kinds of definitions. An **intensional definition** is a definition that describes the intension of a concept by stating the superordinate concept and the delimiting characteristics. The definition of delimiting characteristic in the section on Concepts is an example of an intensional definition. An **extensional definition** is a definition of a concept formed by enumerating its subordinate concepts under one criterion of subdivision. The definition of relation in the glossary is an example of an extensional definition. Note, both kinds of definitions depend on knowing the definitions of other concepts in order to fully understand the concept under study.

C. Data

This section contains a description of the connection between data and terminology. A datum is defined as a kind of designation.

1. Values

A **value** is a concept with an equality operation defined.

Any concept may have an equality operation defined for it. For a set of values, the same equality operation is sometimes defined for the entire set, and this leads to the construction of datatypes. See ISO/IEC 11404 – *General purpose datatypes*. Assigning an equality operation to a concept implies that if, say, two people say they have that concept, a determination of equality between them can be made. For example, two people agree they have the same gender. This operation may be different depending on the situation. In fact, more than one measure of equality can be defined for any given concept. See Example 3.

EXAMPLE 3: Consider the natural number "seventeen". It is a concept, and its extension is all situations of 17 objects. Equality may be defined as it is commonly understood for natural numbers. Another way to define equality for natural numbers, including "seventeen", is to ask if the number is even or odd. In this situation, all odd numbers are equal, and all even numbers are equal.

2. Datum

A **datum** is a designation of a value.

A fundamental requirement for a datum is that it can be copied. In Information Technology, the need for copying happens all the time in data processing. The only way to know a datum has been faithfully copied is to compare the copy to the original. The comparison determines whether equality is satisfied (and the copy is faithful). Therefore, a datum must satisfy the equality <11404> property.

EXAMPLE 4: M for married, as in some person is married. Married is a value, since marriage is a social and legal status controlled by the state. Equality may be determined by referencing the meaning in common law.

A datum is often generated in some context, and this context is what connects it to Figure 6 and to the connection between concepts and objects. Suppose we consider the object Donald Trump, and we determine he has orange hair. Donald Trump is an object, and we can find a concept for which he is in its extension. We know, for instance, he is president of the United States, so he is in the extension of the concept of presidents of the US. This concept has characteristics, and one of them is hair color (of a president). For argument's sake, suppose all the possible hair colors of presidents are Orange, Gray, and Brown. Thus, each president (each object in the extension of the concept presidents of the US) has one of the possible hair colors. Washington's hair color was Gray, and Obama's is Brown. In each case, the appropriate one must be determined. So, the possible hair colors are determinants, and they are possible properties of the characteristic hair color.

Now, given that the hair colors Orange, Gray, and Brown are all the ones possible, every president is assigned one and only one color. Assuming the extension of a concept is a set, this means hair color forms a partition of the extension of presidents of the US. Each class in the partition is defined by one of the properties. In this case, there are 3 classes: Orange, Gray, and Brown. No president belongs to more than one class, and every president belongs to at least one. This characterizes a partition.

When we determine the hair color of a president, we might want to record that, so we assign signifiers to each of the possible properties: for instance, o for Orange, g for Gray, and b for Brown, and through this assignment we create designations called codes. Again, by observation, we have a way to decide if two presidents have the same hair color, and this is based on light reflectivity and color reception in the judge's eyes. So, there is an equality operation for each of these properties. This means each of the properties is a value, each code is a designation, and when we assign a hair color and write down a signifier representing the determination, a datum is produced.

Here, the equality operation for each property (value) serves as an equivalence relation for the partition. Two objects (presidents) have the same property and are in the same class in the partition if and only if they have equal hair color. This equality is assessed through the equality operation previously defined.

EXAMPLE 5: Example of a partition of people based on marital status
Concept = people of the UK
Characteristic = marital status
Partition = {single, married, divorced, widowed}, where "single" means never married and the rest correspond to their usual meanings. The signifiers S, M, D, and W designate these concepts, respectively.

EXAMPLE 6: Second example of a partition of people based on marital status
Concept = people of the UK
Characteristic = marital status
Partition = {single, married}, where "single" means not married and married takes its usual meaning. The signs S and M designate these concepts, respectively. The purpose of the example is to show that more than one partition may apply to a characteristic of a concept.

EXAMPLE 7: Example of a partition of gambling casino games based on probability of winning
Concept = gambling casino games
Characteristic = probability of winning
Partition = $\{x \mid 0 < x \leq 1\}$ (the set of all numbers, x, such that x is greater than zero and less than or equal to one), where x is a probability. The signs are the numeric strings that designate the numbers, to some agreed upon precision, fixing the lengths of the strings.

D. Variables and Aggregates

Variables and aggregates are determinable (in the sense previously described), and therefore are characteristics of some concepts. Variables are mostly characteristics for general concepts, and aggregates are mostly characteristics of individual ones. This corresponds to the notion that a variable is a mapping between some collection of units (the extension of the general concept for which the variable is a characteristic) to a set of values. An aggregate does the same, except the concept is an individual one, so there is one unit – the aggregate.

There are some exceptions. In socio-economic statistics, a household income is the sum (an aggregation) of the incomes of each of the individuals in that household. This aggregate applies to a general concept (i.e., households).

Table 1 shows how the terminological constructs described correspond to common notions about data found in socio-economic data.

Socio-Economic Data	**Terminology**
Unit Type or Universe	Concept
Microdata	General concept
Macrodata	Individual concept
Frame	Extension
Variable or aggregate	Characteristic
Unit	Object (in the extension of the concept)
Observation (or estimation)	Property

Table 1: Socio-Economic Data versus Terminology

E. Variable Cascade

In DDI - CDI, the variable cascade is the way the descriptions of variables is managed. The main purpose of the cascade is to increase the reuse of metadata. The features defined at each level of the cascade don't depend on features at any of the lower levels. Because of this, the descriptions at each level are reusable.

The cascade consists of four levels, each level corresponding to an ever-increasing descriptive detail. The levels in the cascade are

- Concept
- Conceptual variable
- Represented variable
- Instance variable

The names of the levels indicate to the user what the main focus of the description is at each. The Concept and Conceptual Variable provide details about the concepts employed. The Represented Variable and Instance Variable provide the details about the codes, characters, and numbers representing the concepts at the higher levels.

We will describe these levels and show how they fit into the terminological approach in the following sections. In tables in each section, we illustrate the approach with two examples. The attributes are taken from the class diagram of DDI - CDI. We only illustrate the attributes at each level. The inherited ones from the level above are assumed.

1. Concept

The variables about some subject share that subject as common among them all. For example, all variables in use in data sets in a research library about marital status share that concept among them all.

There may be little in common about the marital status as measured in each variable, but marital status itself – the fact there are statuses across societies or cultures – is a common characteristic. The concept expressing this commonality is the purpose of this highest level.

The concept at this level is very generic, because it must account for all possible variations of the more specialized versions attached to each variable that makes use of it.

Concept

ID	***Name***	***Definition***
1	Marital status	Category of current marital arrangement
2	Age	Whole number of years of operation

2. Conceptual Variable

The Conceptual Variable is the level at which most of the concepts used to describe a variable are applied. The main concepts are the determinable associated with a variable and the possible determinants. In our marital status example, the main concepts are:

- Determinable: marital status
  - The specialized nature of this concept is that it is applied to people living in the US (for instance)
- Determinants: kinds of marital status
  - Single
  - Married
  - Divorced
  - Widowed

This example illustrates that at the conceptual variable, the determinable and determinants are concepts. Suitable determinants form an extensional definition for the determinable. In our case, single, married, divorced, and widowed do form an extensional definition for marital status. The determinants are known as substantive values in DDI - CDI.

Additional concepts are those associated with missing data. These are known as sentinel values. The two most important, ones that the statistical packages use, are “missing” and “refused”. There might be others, depending on processing needs.

Conceptual Variable (Links to Concept)

Name	Marital Status	Age
Concept	Concept #1	Concept #2
Unit type	Person	Business establishment

Substantive Conceptual Domain	Single Married Divorced Widowed	Count (of years), top coded at 25
Sentinel Conceptual Domain	Missing Refused	Missing Refused

In this table, the names of the categories for marital status in the substantive conceptual domain are there in place of actual concepts. The only way to write down a concept is either through providing a definition or providing an unambiguous term or word denoting it.

3. Represented Variable

The main addition at the Represented Variable level is the signifiers for the determinants, or substantive categories. Assigning signifiers to concepts turns them into designations. So, in our example, we might end up with the following designations:

- <s, single>
- <m, married>
- <d, divorced>
- <w, widowed>

The set of these designations is a substantive value domain. As discussed, the underlying concepts form an extensional definition for the determinable, the concept associated with the variable. So, these determinant concepts are associated with the subject matter of the variable, not with processing. A substantive value domain can be used by many Represented Variables, so it is important to identify and manage them.

Represented Variable (Inherits from Conceptual Variable)

Name	Marital Status	Age
Universe	Deer Hunters	Gun Shops
Substantive Value Domain	<s, Single> <m, Married> <d, Divorced> <w, Widowed>	Count (of years) represented with 2-digit Arabic numeral
Unit of Measure	N/A	years

Intended Datatype family	Nominal	Quantitative discrete

The Intended Datatype Family attribute above needs some explanation. The interpretation of datatypes in this document is contained in the international standard *ISO/IEC 11404 – General purpose datatypes.* This standard provides 2 ideas that are central to understanding the Intended Datatype Family attribute here.

First, each datatype is defined through 3 ideas:

1. ***Value Space*** – the specific values the datatype covers, which are contained in the Substantive Value Domain for any variable. In the case of the Marital Status variable in the example above, the elements listed under the Substantive Value Domain make up the Value Space.
2. ***Characterizing Operations*** – the operations that distinguish one datatype from another. For example, an area allows for a perimeter and an area calculation, but a distance is just a linear measure.
3. ***Axioms*** – the rules one may assume the data obey. Confusingly, these are called *properties* in ISO/IEC 11404. One 11404 property all datatypes have is equality, and this, as we've seen above, is the defining characteristic that distinguishes data from terms or designations in general. Another example of an axiom is whether data are bounded or not. A finite list of numbers must be bounded, but the list of positive integers has no largest value. It is unbounded at the top.

The combination of axioms and characterizing operations for a value space determine a computational model associated with the data defined in the value space. By alluding to the need for defining equality for a concept defining a datum, we implicitly call into play an underlying computational model. Data are used for computation; and specifying the limits of that computation is what a datatype is for.

Second, ISO/IEC 11404 does not, and cannot, say what the value space is for the descriptions of datatypes contained there. So, the descriptions are generalized, and they address the axioms and characterizing operations that each kind of datatype, or datatype family, must have.

For instance, datatypes describing the marital status values above and genders (the set of gender categories and codes: {<m, male>, <f, female>}) have the same axioms and characterizing operations. The only difference is the value space. As such, these datatypes are both members of the same datatype family, namely the statistical type called nominal.

Nominal, Ordinal, Interval, Ratio, Quantitative, Qualitative, Discrete, and Continuous are names of datatype families typically used in the statistics. In the examples above, we make use of these.

4. Instance Variable

Moving further down the chain to data, we get to the Instance Variable. An Instance Variable is intended to be a variable used in a data set. For each data set, new Instance Variables are created.

The main addition in specificity is turning the sentinel categories into designations. Further, the list of sentinel values (designations) are managed in one set, the sentinel value domain. Separating the

substantive and sentinel value domains eases the burden on metadata management. Changes needed in one kind of value domain do not affect the other.

An example of the designations in a sentinel value domain is:

- <m, missing>
- <r, refused>

Since, the Instance Variable is associated with data in a data set, then the datatype of the data for that variable is necessary information as well.

Instance Variable (Inherits from Represented Variable)

Name	Marital Status	Age1
Population	US Deer Hunters in 2019	US Gun Shops in 2019
Sentinel Value Domain	<1, Missing> <2, Refused>	<1, Missing> <2, Refused>
Function	demographic	establishment
Physical Datatype	1-character	2-integer

The codes used to designate the sentinel categories are often determined by each statistical package. This topic will not be addressed in detail here.

The Physical Datatype addresses the kind of data as written on a file. The value $2.60 (two dollars and sixty cents) is often written as a real number with 2 decimal places. But monetary amounts don't follow all the rules for real numbers. The amounts at the third decimal place or after are truncated. The values are not rounded, as real numbers will be. This has an effect on computations, as the following example illustrates:

Take the average of $1.50, $1.30, and $1.00. The arithmetic average is $1.2666. The rounded real number average is $1.27, and the monetary, or scaled number, rounded average is $1.26. The reason is the fractional penny is dropped in the scaled situation. And the rules for scaled numbers correspond to how banks handle money.

Therefore, the physical datatype is often just an approximation of what is needed to describe values. Instead, it corresponds to how the values are written in a file. The actual use of the values depends more on the Intended Datatype at the Represented Variable level

F. Detailed Documentation for Foundational Metadata in DDI - CDI

Detailed documentation at for the Foundational Metadata in DDI – CDI can be found in this package in the folder: `\DDI-CDI Public Review 1\2 Model\Field-Level Doc\`.

III. Data Description

A. Introduction

The DDI-CDI model is defined as a Unified Modeling Language (UML) model. Figure 8 below shows a core portion of that model. The elements (classes) of the model appear as boxes with a name at the top and a list of properties below the name. Properties, listed in the bottom half of the box for the class, contain the payload of the class. Sometimes the value of a property is complex. The "definition" property of a Concept, for example, has the datatype of "InternationalStructuredString", which will have a text string, but also other properties such as whether it is translated and from what language it is translated. This complexity is the result of 20 years of incorporation of use cases into the model.

Classes may also have associations with other identifiable classes. In the diagram below a Datum has a simple association named "denotes" with a ConceptualValue. This relationship is read as "a Datum denotes a ConceptualValue". It is displayed in the diagram as an arrow that indicates the order in which the association is to be read. Classes that can be the target (object) of an association have a unique identifier and are reusable. The target end is indicated by an open arrowhead.

Some classes inherit from others. A ValueString inherits (is a type of) from an InstanceValue. This is indicated by the filled-in triangular arrowhead on the parent end.

Some associations indicate containership. A ConceptSystem aggregates (has) a set of Concepts. This is indicated by the diamond on the containing end of the relationship line and is read as "a ConceptSystem has Concept".

A Datum populates a cell of a dataset, database table etc.[4] through an InstanceValue. The Datum links some conceptual value to a physical representation. In the diagram below a ValueString is a physical representation. It future versions of the model, the physical representation could be an image, a sound byte or some other digital representation. Introduction of the ConceptualValue allows for the description of multiple representations in perhaps multiple platforms of the same measurement. A height, for example could be recorded as a decimal string or a binary string.

Physical structures (like files) are made up of DataPoints, each of which contains one InstanceValue.

In this model, InstanceVariables constrain DataPoints and Datums. They are no longer restricted to describing a column in a table.

The general idea in DDI-CDI is to be able to attach metadata at the "cellular" level, rather than at the structural level, and to allow those "cells" to be arranged into different structures without loss of descriptive information.

(For more information about how UML diagrams are used in DDI – CDI, please see the document "DDI Cross Domain Integration: Architecture and Alignment with Other Standards" in this package.)

This is how Datum is defined in the DDI4 prototype documentation and GSIM.docx

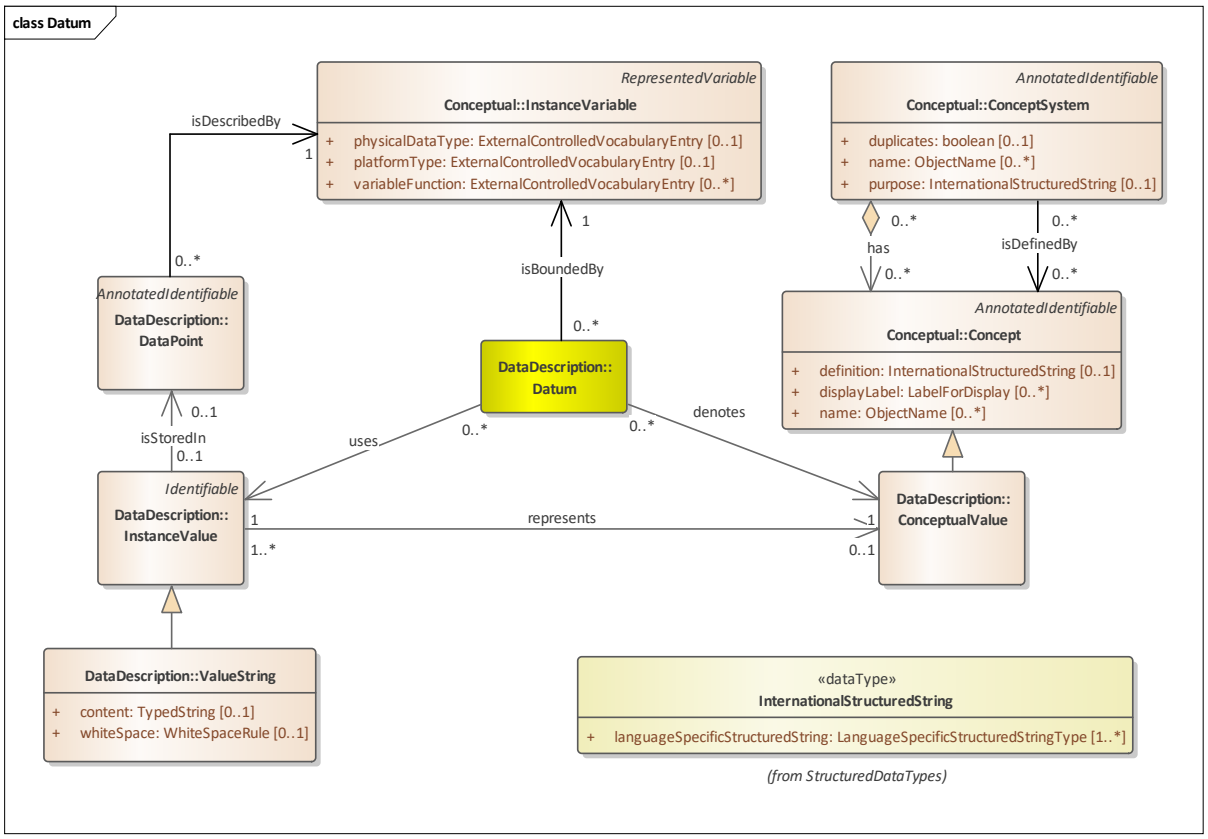


Figure 8: Datum in the DDI-CDI Model

B. Detailed Documentation for Foundational Metadata in DDI - CDI

Detailed documentation at for Data Description model in DDI – CDI can be found in this package in the folder: `\DDI-CDI Public Review 1\2 Model\Field-Level Doc\`.

C. Scope

The DDI-CDI Data Description provides the basis for describing a broad range of data structures using a core set of metadata elements. The model separates data structure and content in such a way as to allow data to be structured flexibly.

This section describes the general approach which DDI-CDI is taking, before going through the details for a selected set of data structures. The goal for DDI-CDI is to describe various data structures, both legacy structures such as rectangular data sets, multi-dimensional data, and event data, but also new ones like data streams or data lakes. The approach is independent of any specific domain or discipline, as similar data structures are used broadly in a range of research settings.

The model has structures for documenting different data structures and the transformations between them.

Data structures are a way to organize data in an organized way in order to be processed by software programs. The current DDI-CDI model can be used to describe data from different data structures using a Datum-based approach. This approach involves describing each “cell” in a granular fashion, such that the same values can be recognized when occurring in differently structured data sets.

The following structures are covered:

Wide Data: Traditional rectangular unit record data sets. Each record has a unit identifier and a set of measures for the same unit.

Long Data: Each record has a unit identifier and a set of measures but there may be multiple records for any given unit. The structure is used for many different data types, for example event data and spell data.

Multi-Dimensional Data: Data in which observations are identified using a set of dimensions. Examples are multi-dimensional cubes and time series. (Note that support is provided for time-series-specific constructs to support some legacy systems which are not based around the manipulation of multi-dimensional data “cubes”.)

Key-Value Data: A set of measures, each paired with an identifier, suited to describing No SQL and Big Data systems.

Each of the four data structure types - Wide, Long, Dimensional, and Key-Value - are structured in slightly different ways but share some common features. Before going into how each of them can be structured a set of related core components will be presented, applicable across the range of data structures described.

This chapter describes the DDI-CDI approach to each of the above-mentioned data structures according to the following outline:

A. Introduction

B. Detailed Documentation for Foundational Metadata in DDI – CDI

C. Scope

D. Basic Concepts
   1. Variables and Values
   2. Keys
   3. Data Structure Components

E. Wide Format (Unit Record Data Structure)
   1. Example
   2. Discussion of Structure and Diagrams – Wide

F. Long Data Format
   1. Example
   2. Discussion of structure and diagrams – Long

G. Multi-Dimensional Format
   1. Example
   2. Discussion of structure and diagrams – Dimensional

H. Key-Value Format
   1. Example
   2. Discussion of structure and diagrams – Key-Value

H. Physical Data Set (Wide Format)

I. Transformations between Formats/Examples
   1. Wide and Long: Correspondence between Unit record data and data in a Long format
   2. Wide and dimensional: Unit record data tabulated into an aggregate data Cube
   3. Long and Dimensional: Dimensional data represented in a Long data format
   4. Key-Value and Wide: Key-Value Stores in RAIRD
   5. Time Series
   6. Key-Value Stores and Streams

D. Basic Concepts

Before explaining about the four data structure types some basic shared concepts require explanation.

1. Variables and Values

Consider this portion of the model:

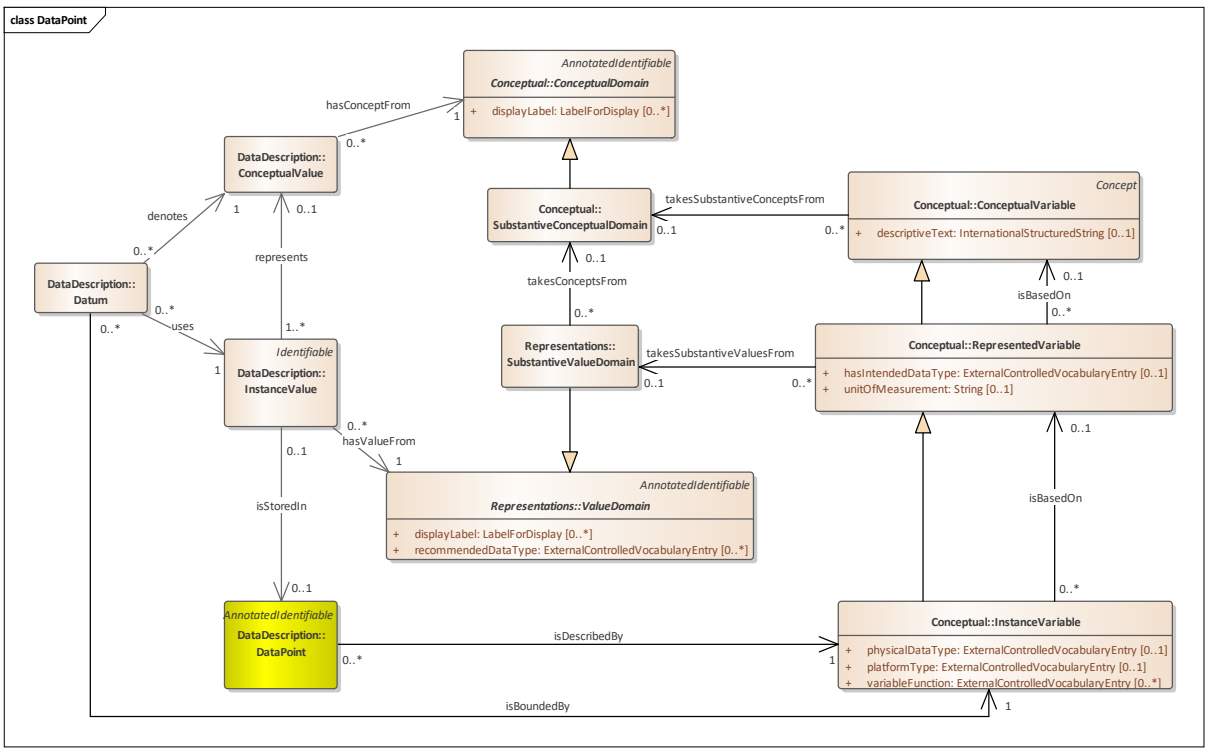


Figure 9: DataPoint in the model

To the right in Figure 9 above the variable structure that provides meaning to the data is displayed. (The variable structure is often referred to as the Variable Cascade, as described earlier in this document in the section on Foundational Metadata.)

The Variable Cascade facilitates distinctions between Instance Variables (the variable as in the dataset), Represented Variables (the reusable components of a variable) and Conceptual Variables that expresses the conceptual basis of a variable.

In the middle of the figure domains are displayed, which provide representations and contexts for the data values. The SubstantiveConceptualDomain specifies the set of valid concepts for the ConceptualVariable, while the SubstantiveValueDomain specifies the set of values for the corresponding InstanceValues. ConceptualDomain and ValueDomain are abstract classes of which SubstantiveConceptualDomain and SubstantiveValueDomain are sub-classes.

The left part of the figure displays how data values are modeled. The DataPoint represents a 'cell' in a data structure that stores the InstanceValue, which is the actual representation of the observation or value in a data set. Datum is a mediator between the InstanceValue and the ConceptualValue, which is a more generic and reusable value format.

The DDI 4 Data Description is based at the core on the description of a single datum associated with a data cell (a DataPoint). One example of a DataPoint is a single cell in a rectangular table. DataPoints, in turn, are organized into structures. (This will be described further below.)

Earlier versions of DDI (e.g., the DDI Codebook and DDI Lifecycle specifications) allowed for the description of two structures, records and NCubes. The record structure was defined in terms of a list of variables and did not allow for the description of individual cells. The NCube structure was an assembly of variables, to allow for the process of tabulation to be recognized. DDI – CDI takes a more granular and more generalized approach.

DDI-CDI also explicitly models the conceptual value that is represented by the symbol in the DataPoint along with the Unit measured by the value associated with a DataPoint. The DDI-CDI approach allows for the explicit description of different representations of the same measured value. This is called Instance Value in the model. A measurement (the captured value) of Marie's (the Unit) longevity (the type of measure) at a specific time might, for example, be represented in Arabic numerals, roman numerals, or words. In our unit record example Marie's longevity is recorded in Arabic numerals as "73.7". The measurement might be contained in different software packages that have subtly different representations of a number. All of these represent the conceptual (measured) value of the longevity but use different symbols to do so.

Following GSIM, DDI-CDI includes the elements *Datum*, and *DataPoint*. In the DDI-CDI model, a Datum connects a conceptual value (a type of Concept) with a representation (a perceivable symbol, the instance value). In the case of Joe's height taken on 2019-07-16, a Datum would link the conceptual value of that height to a specific sequence of characters (e.g. "183,5" cm). Another Datum might link

that same conceptual value to a different sequence (“183.5” cm). Both instance values stand for the same measurement but use different physical symbols.

A DataPoint is a kind of container for the representation of a value, the instance value. Think of a cell in a spreadsheet. Its column corresponds to an InstanceVariable. Its row corresponds to a Unit. It may have content or it may be empty. Changing the format of the cell changes the representation (instance value), but the underlying value (conceptual value) remains the same.

This allows the single Datum to be be ‘followed’ across different data structures. This differs from approaches used in many other similat models, including other DDI products (e.g., DDI Codebook, DDI Lifecycle) where some of this information was attached at a higher level (typically the data set or record). DDI-CDI is a little more explicit than GSIM in describing the conceptual value and its representation, the instance value. As an information model, GSIM does not deal with the details of physical representation of data. DDI-CDI needs to be able to describe representation in more detail.

A Datum populates a cell of a dataset, database table etc. The general idea in DDI-CDI is to be able to attach all necessary metadata to the single Datum so that it can be ‘followed’ across different data structures. This differs from some other approaches used in other DDI products (DDI Codebook, DDI Lifecycle) where some of this information was attached at a higher level (eg, the data set or record).

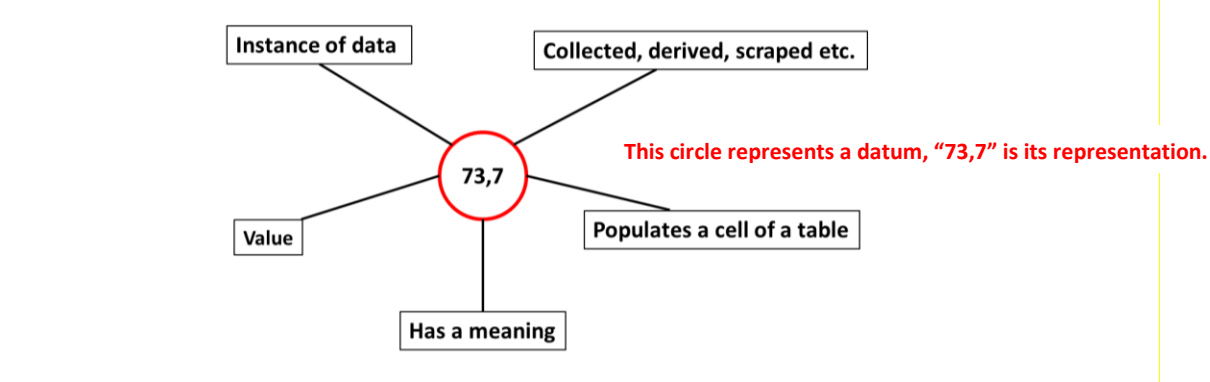


Figure 10: Datum and its connotations

2. Keys

Another central concept for Data Description is that of the Key. In the model a Key is used for the identification of data and may comprise a set of Key members or be a unitary value. Figure 11 below shows how an InstanceValue is linked to a Unit (an individual or object of interest) via a Key that identifies the DataPoint where the InstanceValue is stored.

A Key is defined as a collection of data instances that uniquely identify one or more data points. A KeyMember is a single data instance that is a part of an aggregate key.

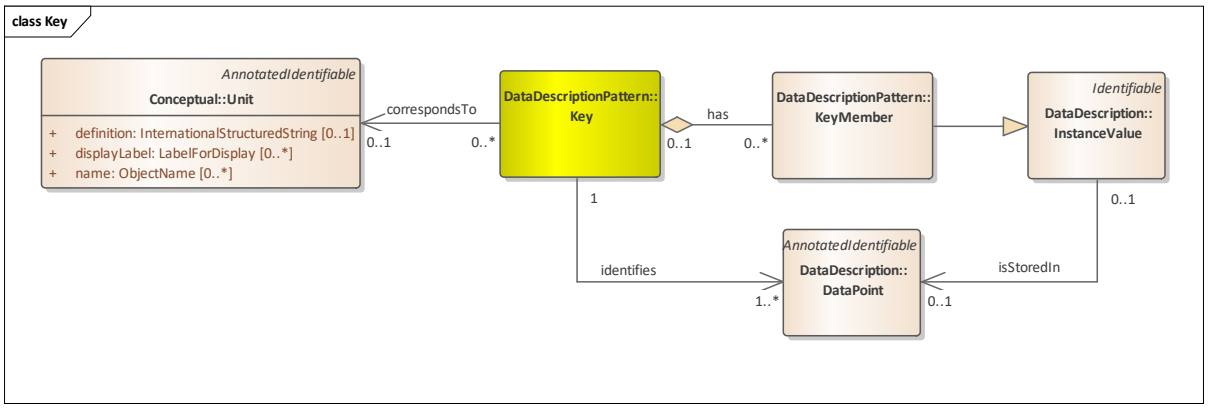


Figure 11: Key

3. Data Structure Components

A third feature are DataStructureComponents, allowing the RepresentedVariables to take different roles. Each data structure type - Wide, Long, Multi-Dimensional and Key-Value - has its own set of DataStructureComponents, with some being common across data structure types. The components which reflect roles are the IdentifierComponent, AttributeComponent and MeasureComponent. The roles allow a RepresentedVariable to serve as a Measure in one context and as an Identifier in a different context, for example. This will be detailed below in the description of the different data structure types.

Roles allow users to assign different functions to variables according to their context of use. Roles are not inherent in variables but can be imposed on them. In DDI-CDI there are currently three roles:

- **Identifier** - An identifier role that serves to differentiate one record from another. More than one variable may be used in combination to produce a compound identifier.
- **Measure** – Variables tagged with the measure role represent the values of interest.
- **Attribute** – The attribute role serves to provide information about the measures of interest. Variables might, for example, describe the conditions of a measurement. This way attributes can be used to link metadata or paradata to the Measure of interest.

A variable may take on different roles in different contexts.

E. Wide Format (Unit Record Data Structure)

1. Example

A Unit Record data table, as shown in Figure 12, is a common way to organize data. Each record has a set of observations about a single unit. The record has a unit identifier and a set of measures and/or descriptors which are the same for each unit. The unit identifier can be used as an identifier for the record, because each unit has only one. This structure is also referred to as a rectangular data file.

PersonID	Sex	Born	Died	RefArea	Longevity
Marie	Female	3.3.1932	12.1.2005	Newport	73.7
Henry	Male	8.1.1929	6.2.2008	Cardiff	78.8
etc.					

Figure 12: Unit record data table

The objects of the Wide format Unit Record data table are UnitDataRecords, Variables and InstanceValues. In the Wide format the rows correspond to each unit record, which is a set of InstanceValues for one entity. The columns correspond to each variable measure or categorization. Cell entries are InstanceValues.

A cell in the Unit record table is an intersection between a column representing a variable and a row representing a measurement unit. See for example '8.1.1929' in Figure 13 (yellow highlighting).

Each cell of the table contains an InstanceValue. 'Marie' and 'Henry' (green highlighting) are identifiers for each of the records. 'Sex', 'Longevity' etc. are variables (blue) and 'Female' and '78.8' are example of InstanceValues (red).

PersonID	Sex	Born	Died	RefArea	Longevity
Marie	Female	3.3.1932	12.1.2005	Newport	73.7
Henry	Male	8.1.1929	6.2.2008	Cardiff	78.8
etc.					

Figure 13: Wide format object

The WideDataSet contains DataPoints, all the 'cells' in the table. Columns are variables, and each row contain the DataPoints for one Unit. Some of the DataPoints contain values keys that identify the DataPoints common to an individual row of the table. A WideKey can have more than one Member - e.g. more DataPoints which act as identifiers. This will be further explained below.

2. Discussion of Structure and Diagrams – Wide

A Wide table row is further structured by three DataStructureComponents types:

- IdentifierComponents - the DataPoints which serve to identify the row.
- MeasurementComponents - the DataPoints in each row which contain the measures of interest.

- AttributeComponents - DataPoints which provide context for the MeasureComponents.

RepresentedVariables provide SubstantiveValues for a WideKeyMember.

In the example dataset displayed in Figure 14 below the "PersonID" column contains DataPoints that contain the key values that identify a row and also correspond to a Unit.

The DataPoint in the upper left of the table contains the key value "Marie". That DataPoint identifies the other DataPoints also associated with the person named "Marie", the DataPoints in the first row of the table.

A WideKey can be composed of more than one WideKeyMembers. Our table might have, for instance, have contained another column like "Family" so that we could identify the Marie in a particular family. (This might be important in a data set which had more than one unit named "Marie".)

A row of the table is also further structured by DataStructureComponents.

These are defined by RepresentedVariables, which in turn provide the SubstantiveValueDomain (often a Codelist) for a WideKeyMember.

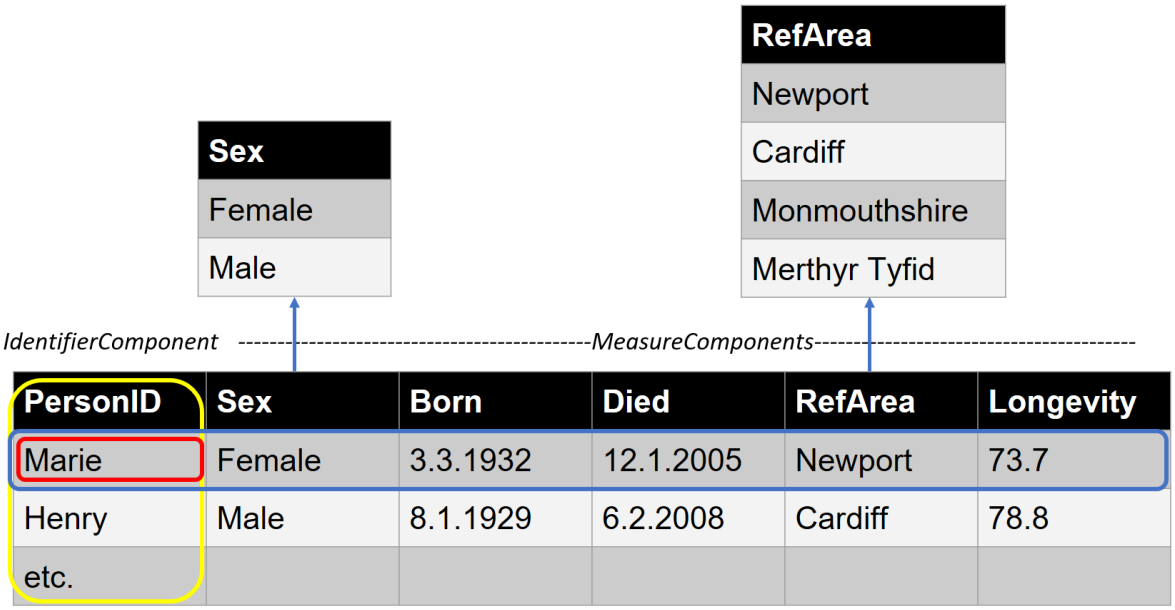


Figure 14: Wide structure

In the figure above, PersonID is an identifier for a person, Sex, Born, Died, and Longevity and RefArea are the measures of interest.

These roles are not fixed. For another purpose, RefArea might be considered an attribute of the measures. Roles are often slightly different when the same data is viewed using different formats (PersonID is the only identifier needed for the Unit Record format above – when expressed in a Long

format, it would be only one needed component of a compound identifier – more than one variable would take on the role of identifier. (See RAIRD example, below).

The diagram in Figure 15 below shows the DDI-CDI classes used to represent unit data in wide format. This is probably the most common layout for data – the traditional table of data as used in many statistical packages and spreadsheet programs. Columns are variables and each row contain the DataPoints for one Unit.

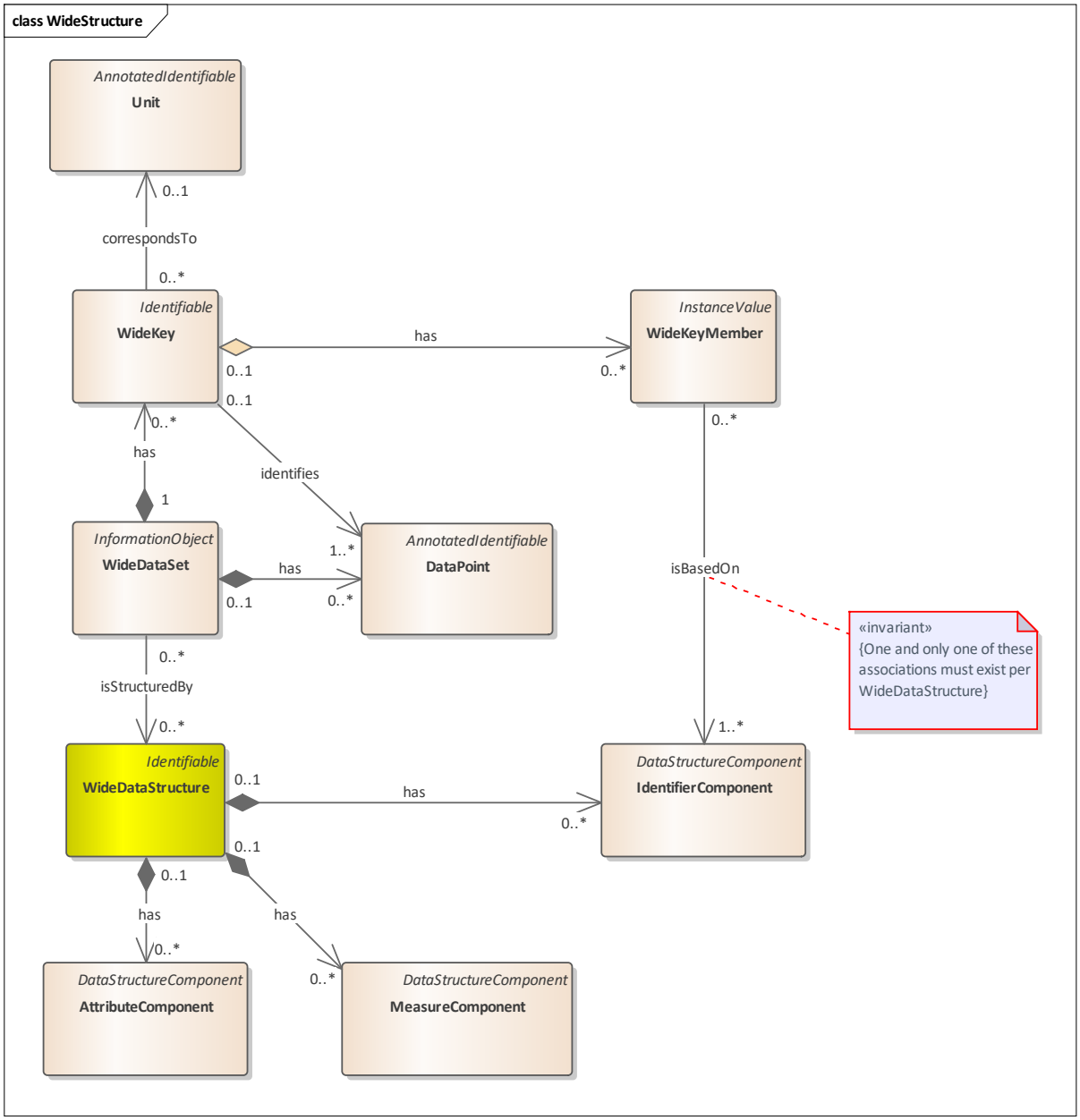


Figure 15: WideStructure

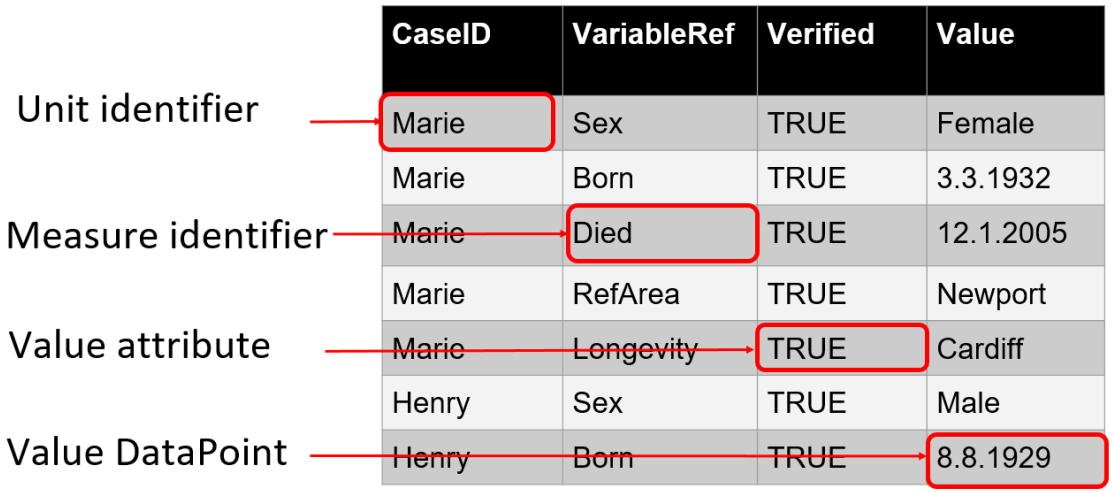
F. Long Data Format

1. Example

The same data as in the Wide example can be expressed in a different format called Long as shown in Figure 16 below. This format is often used to express event data.

In the Long format columns correspond to each kind of object in a Wide (unit record) description. Each row now contains a unit identifier, a variable identifier, and a data point with an instance value.

The rows correspond to each value of each (non-identifying) variable for each Wide record.



CaseID	VariableRef	Verified	Value
Marie	Sex	TRUE	Female
Marie	Born	TRUE	3.3.1932
Marie	Died	TRUE	12.1.2005
Marie	RefArea	TRUE	Newport
Marie	Longevity	TRUE	Cardiff
Henry	Sex	TRUE	Male
Henry	Born	TRUE	8.8.1929

Figure 16: Long format

In pure form, each row of a long structure contains a DataPoint with the value of interest (the instance value) along with identifiers for a unit and a column with a code that identifies the variable (VariableRef above) that associates with the value in the value DataPoint. In the figure above the Value column contains DataPoints with values from more than one variable, Sex, Born, Died, RefArea, and Longevity. Note that there may be many rows for a unit (like for "Marie"). There can also be columns containing attribute values. The "Verified" column is an attribute that indicates whether the value in the Value column has been verified.

CaseID	VariableRef	Value
Marie	Sex	Female
Marie	Born	3.3.1932
Marie	Died	12.1.2005
Marie	RefArea	Newport
Marie	Longevity	73.7
Henry	Born	8.1.1929
Henry	Died	6.2.2008
Etc.		

Here we can see how a complete record for one unit from our Wide example might be represented: each column in the Wide format for a single row becomes a row in the Long format (see Transformations between Data Structures, Examples, below).

2. Discussion of structure and diagrams – Long

The high-level view of the long model is shown below. Each DataPoint in the dataset is based on one of five data structure components. Each component is associated with a RepresentedVariable that can define a column in the tall table.

These perform the following functions:

• IdentifierComponent – one of possibly several components that together identify the Unit associated with the measures and attributes. In the example above this is the CaseID column in Figure 16.

• MeasureComponent – a measure just like in the wide layout. This allows a hybrid wide-tall layout. There is no such column in the example above if there were the values for the Marie rows in Figure 16 would all be the same.

• AttributeComponent – an attribute that annotates the associated measure values. This is the Verified column in Figure 16.

• VariableDescriptorComponent – an indicator of the InstanceVariable in each associated VariableValueComponent DataPoint (see Diagram). This is the VariableRef column above. In the first row the code “Sex” indicates that the value “Female” is associated with the variable named “Sex” used in the Wide table. Note that this component has an association to a specific VariableValueComponent.

• VariableValueComponent – defines a column that has a value associated with the value in the VariableDescriptorComponent. This is the Value column above. The “3.3.1932” is interpreted as the date that Marie was born. This column will have to have a datatype as generic as needed to hold all of the values from the set of variables indicated in the VariableDescriptorComponent. In the example above there is a mix of numeric (Longevity), Date (Born, Died), character (Sex), and geographic codes (RefArea) variables. A character datatype for the associated RepresentedVariable would be required. In many statistical platforms there are tools to reshape data between wide and long format. Many have restrictions that would force all of the measure values to have the same datatype (e.g. all numeric).

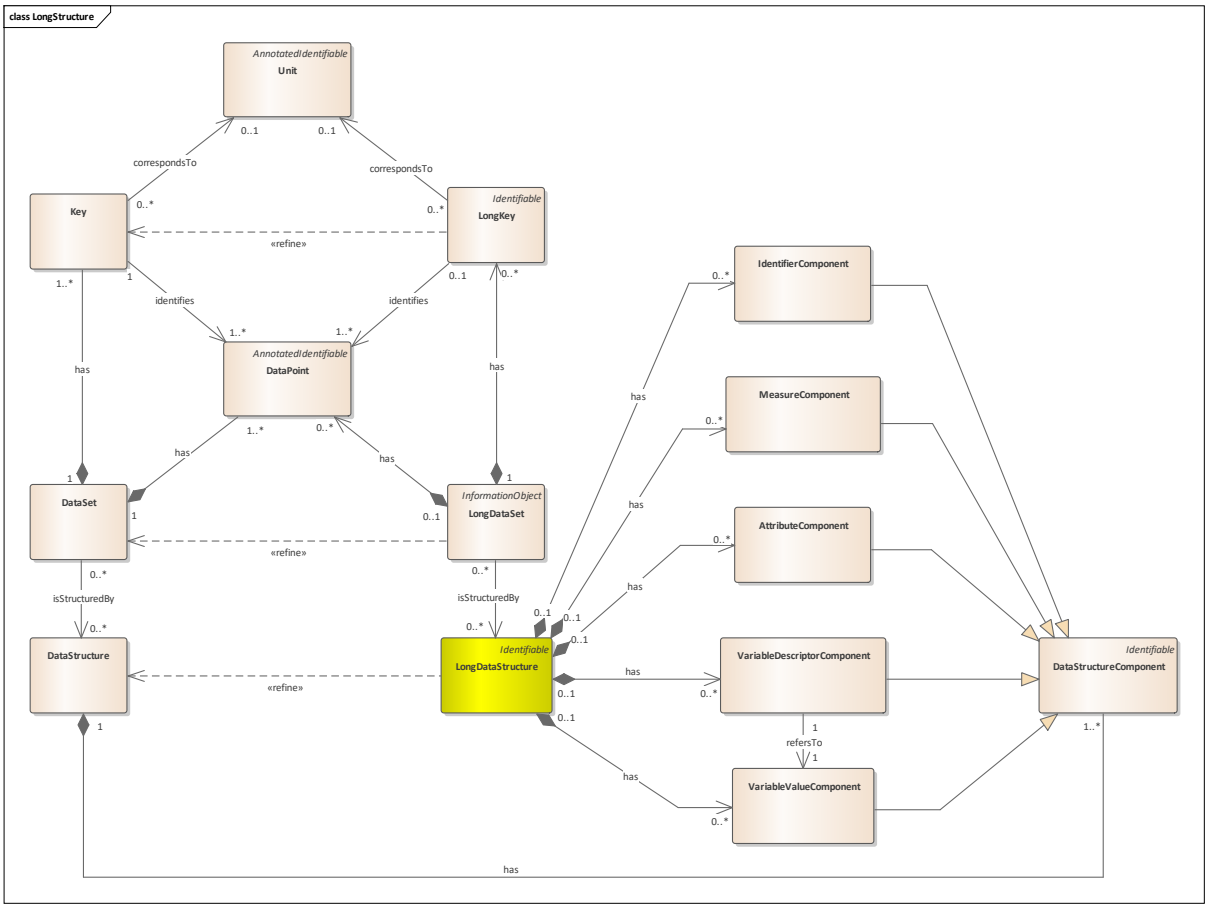


Figure 17: LongStructure – overall diagram

On the right side of Figure 17 we the different data structure components described above are shown.

The left side displays how the Long Key identifies the DataPoint and brings it together with the other components.

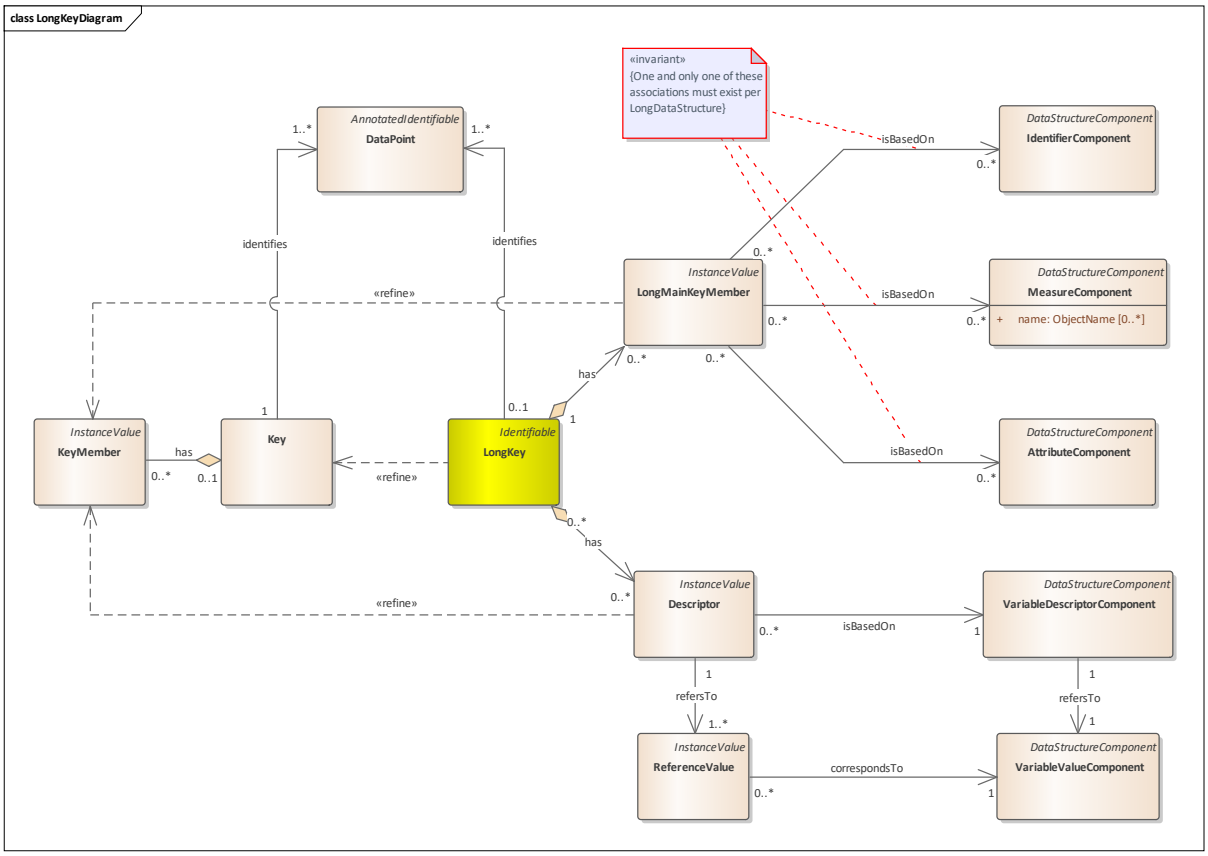
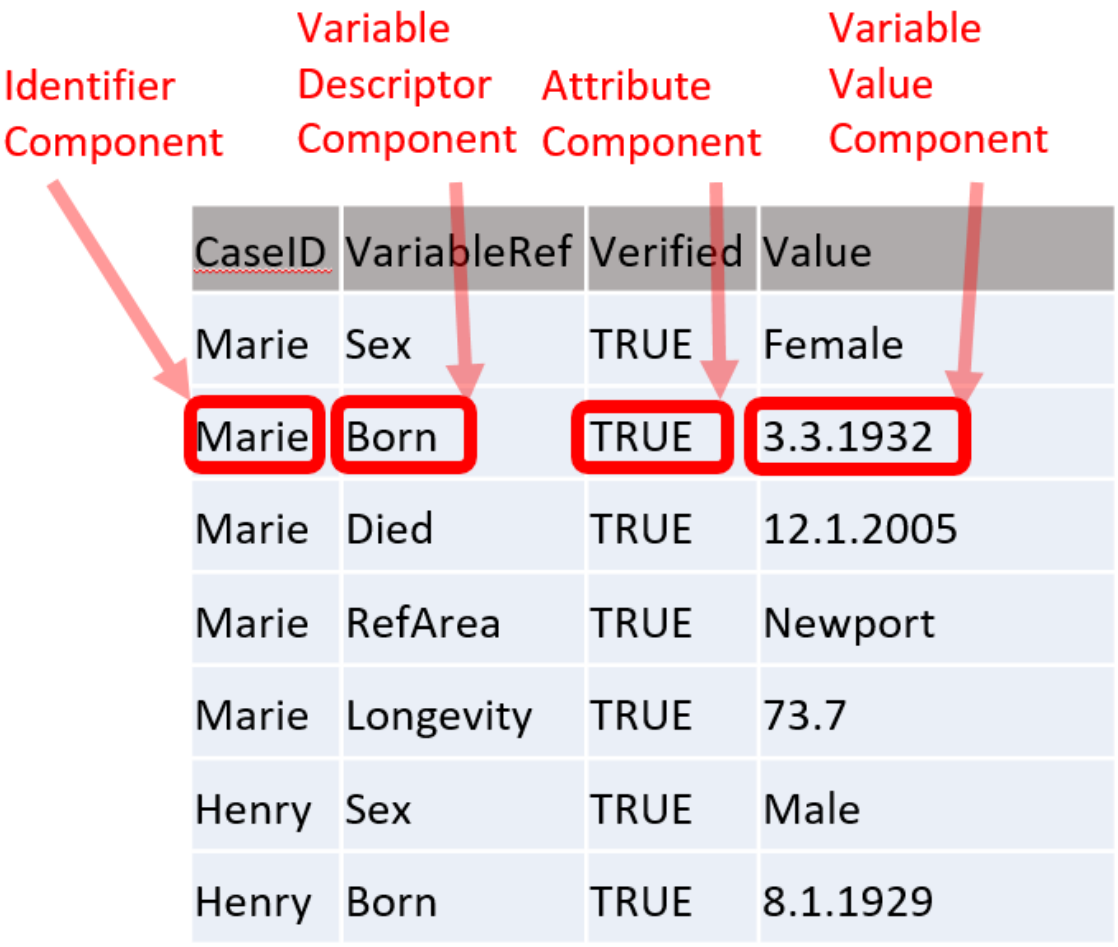


Figure 18: Long format – LongKey diagram

The diagram in Figure 17 conceals some of the complexity involving the association between the LongKey and the LongKeyMember. The LongKey (Figure 18) is actually a composite of LongKeyMembers, each of which is based on one of the five component types. A LongKey could include, for example, two IdentifierComponents, such as Household and personInHousehold.

The long layout brings out the utility of the Datum based approach and the use of keys to describe data. In the long dataset example seen below the values of the “Value” column are in a different conceptual domain in each row. A traditional variable having one conceptual and one value domain makes no sense for the column.

The VariableRef column contains the VariableDescriptorComponents of the compound key describing the InstanceValue in each row of the value column. The column VariableRef itself is a DescriptorVariable that can be described as having codes that point to InstanceVariables. In the highlighted cell in that column “Born” is a code for an InstanceVariable that describes dates of birth. The other two columns are associated with InstanceVariables that could appear in a wide layout. CaseID contains id values each of which is an IdentifierComponent of the compound key. Verified contains AttributeComponents of the key. Together the compound key of “Marie”, “Born”, and “TRUE” provides context for the highlighted InstanceValue of “3.3.1992”. They allow it to associate it with the “3.3.1992” in the “Marie” row of the “Born” column of the wide example table above.



CaseID	VariableRef	Verified	Value
Marie	Sex	TRUE	Female
Marie	Born	TRUE	3.3.1932
Marie	Died	TRUE	12.1.2005
Marie	RefArea	TRUE	Newport
Marie	Longevity	TRUE	73.7
Henry	Sex	TRUE	Male
Henry	Born	TRUE	8.1.1929

Figure 19: Long table components

The VariableDescriptorComponent diagram below shows how the VariableDescriptorComponent relates to other components of the model.

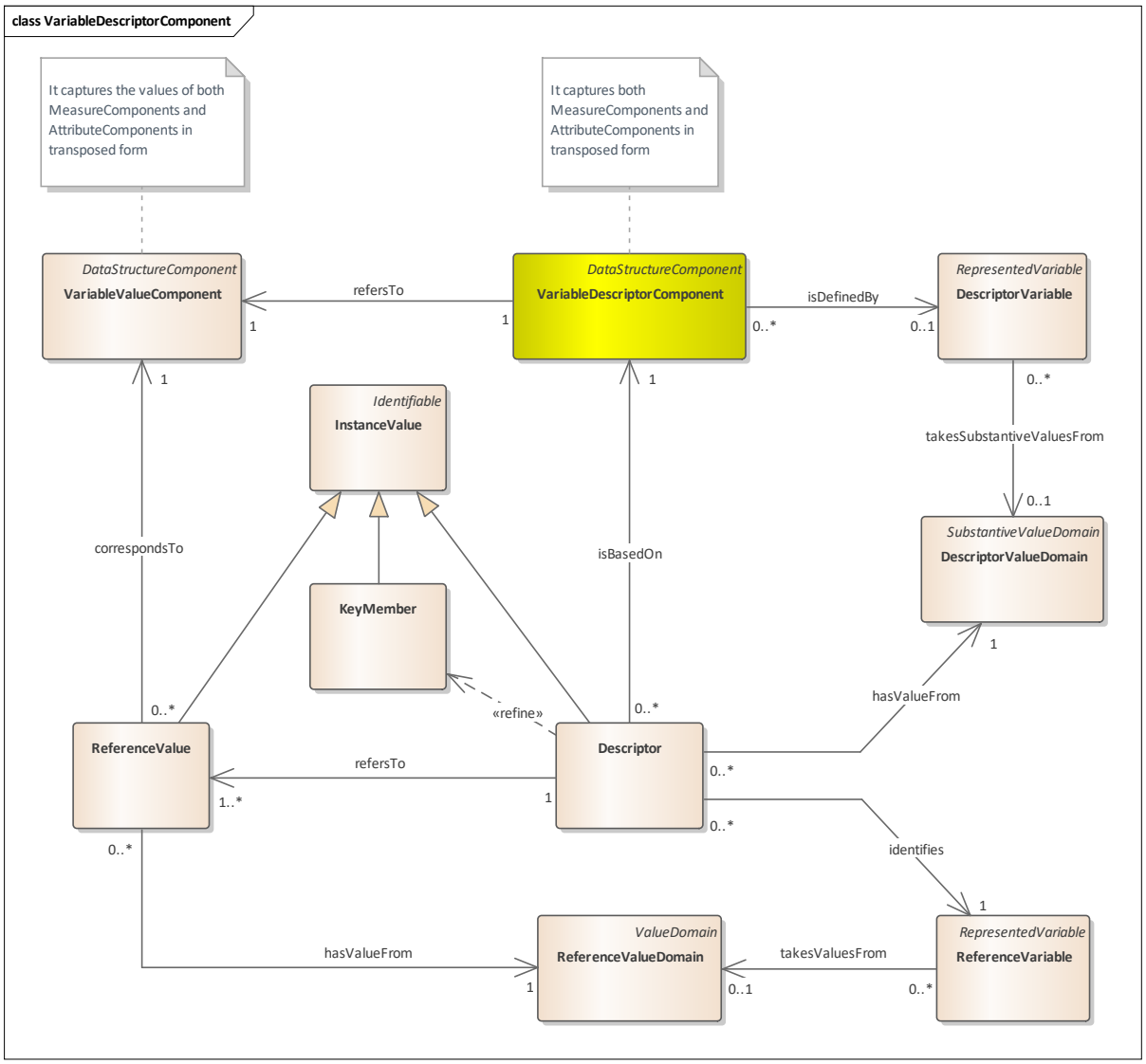


Figure 20: VariableDescriptorComponent diagram

G. Multi-Dimensional Format

1. Example

Sometimes data are presented in dimensional form. In the example below (Figure 21) there are three dimensions: geographic, with Categories of Newport, Cardiff, Monmouthshire, and Merthyr Tydfil.; temporal, with categories like 2004-2006; and gender, with categories of Male and Female. The numeric values in the cells are often aggregates computed on some variable or combination of variables, in this case the mean of longevity. Cells might also contain direct measurements such as with data from an experiment with a factorial design. Dimensional data are commonly displayed in a dimensional table like a pivot table.

	2004 - 2006		2005 - 2007		2006 - 2008	
	Male	Female	Male	Female	Male	Female
Newport	76.7	80.7	77.1	80.9	77.0	81.5
Cardiff	78.7	83.3	78.6	83.7	78.7	83.4
Monmouthshire	76.6	81.3	76.5	81.5	76.6	81.7
Merthyr Tyfid	75.5	79.1	75.5	79.4	74.9	70.6

Figure 21: Dimensional data presented in tabular form

2. Discussion of structure and diagrams – Dimensional

A cube is a multi-dimensional array of cells (DataPoints). Values in the cells may be the result of an aggregate computation or a direct measurement.

At a logical level the structure of the cube is defined by a set of Dimensions (the DimensionalDataStructure in the diagram below).

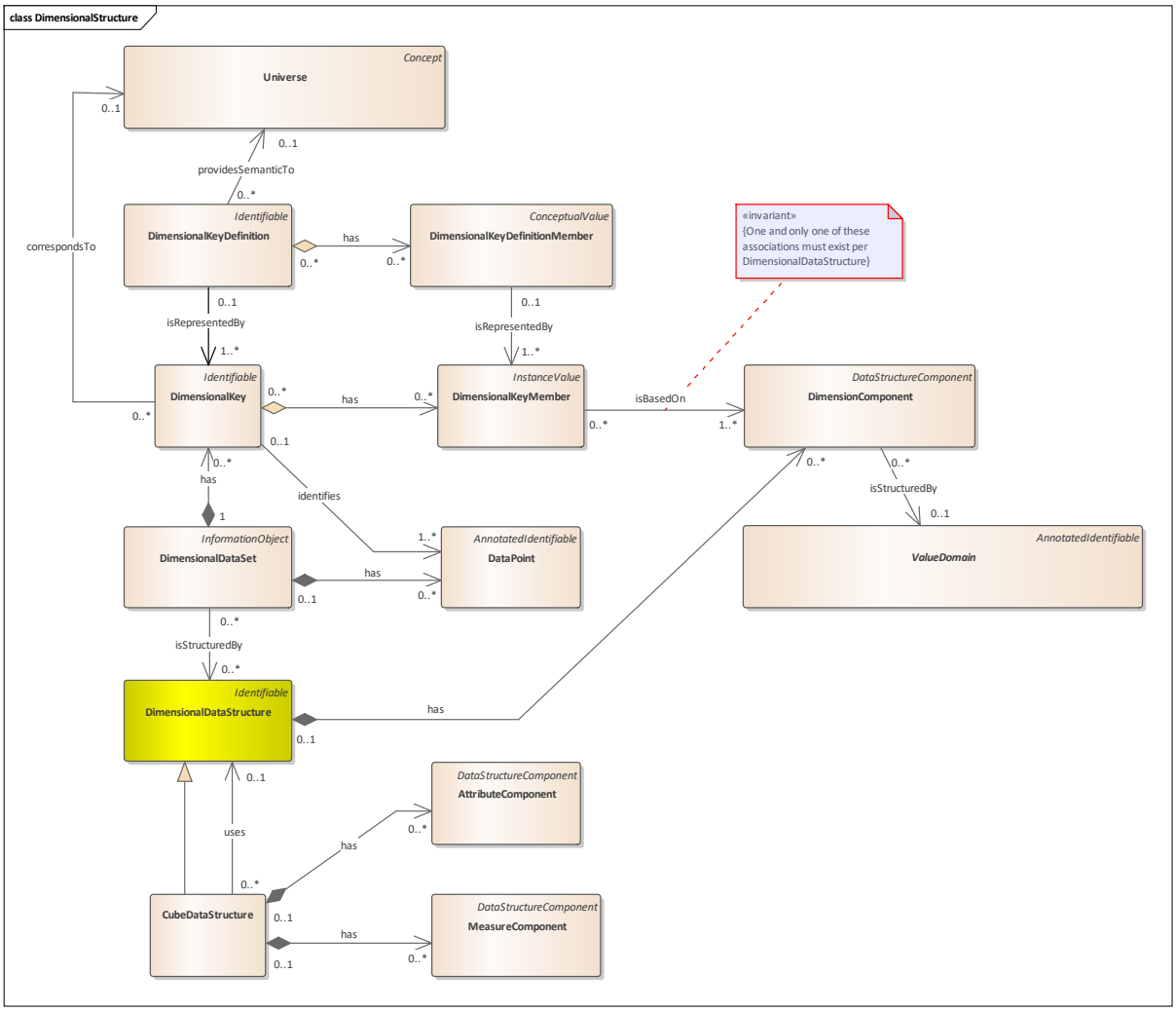


Figure 22: DimensionalStructure - details

Each dimension (DimensionComponent) is, in turn structured by a SubstantiveValueDomain and defined by a RepresentedVariable. The latter also brings along the specification of a Universe and a Concept. A Dimension can be categorical, for example ("Male", "Female"). In this case the SubstantiveValueDomain would consist of a Codelist. Typically cubes containing aggregate data would have primarily (or only) categorical dimensions. A DimensionComponent might also have a described value domain. Experimental data might, for instance, employ an independent variable measured as a real number (e.g. person's weight).

While there may be some underlying continuous variable for a Dimension, a Dimension may often be delineated by discrete dimensional categories. Time, for example, is a continuous measure. In our cube example, though, it has been transformed into a set of three-year categories like 2004-2006. This, along with the other two dimensions (gender, and geography), allows for the delineation of discrete cells in a table. Note that the time periods in this example (Figure 22) overlap.

The DimensionalComponents form the basis for keys. A DimensionalKey is a composite of one value from each SubstantiveValueDomain of a DimensionComponent. This composite DimensionalKey

identifies the location of a DataPoint in the dimensional structure. Our example cube, for example, contains mean longevity data measured on people of Wales. The DataPoint (cell) identified by the key value (2006 – 2008, Female, Newport) is associated with that subset of people.

Partial Keys – in which only a subset of the DimensionalKeyMembers have values specified for them – can be used to refer to regions (or "slices") within the cube. (DDI-CDI does not explicitly model this; it is left to implementations to handle partial Keys if this is useful or required.)

Each DimensionalKeyMember (InstanceValue) of the DimensionalKey is also associated with the concept 'Male' in a ConceptualValueDomain. This would provide meaning for the DimensionalKeyMember in the case of an aggregate data set.

Categories within the Dimension may be additive or not. In our example the geographic areas could be combined to create larger areas. The year range categories could not be combined in a straightforward fashion given that they overlap.

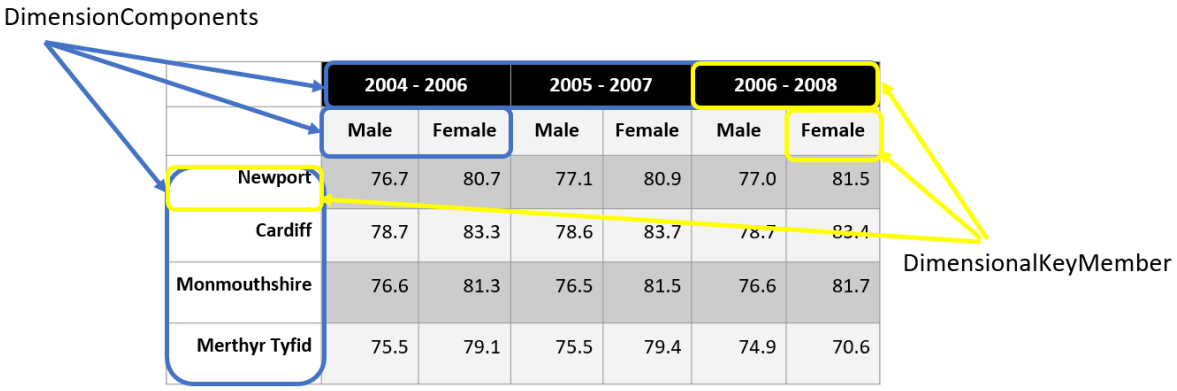


Figure 22: DimensionComponents and DimensionalKeyMembers

In addition to structure a cube has content. The CubeDataStructure also includes a MeasureComponent and an AttributeComponent. The MeasureComponent is defined by a variable for that value.

A QualifiedMeasure as the measure for the whole cube (e.g. mean of longevity), while a ScopedMeasure is for each cell in a cube as its Population narrows the Universe of the Qualified measure

There might also be Attributes associated with each cell in a cube. One example of an attribute might indicate whether the measure for the cell was imputed.

The DDI-CDI model bundles a number of information elements into an Instance Variable. While a cube like our example may have a measure with a single concept, each cell in the cube has a different Population. The upper left cell in the example has a mean of longevity for Males in Newport in 2004-2006. The cell just to the right of it has mean of longevity for Females in Newport in 2004-2006. The DDI-CDI Dimensional model includes the notion of a ScopedMeasure for the InstanceVarriable for each cell in a cube and a QualifiedMeasure as the measure for the whole cube. The ScopedMeasure has a Population which narrows the Universe for the QualifiedMeasure. This diagram (Figure 23) shows how those fit into the model.

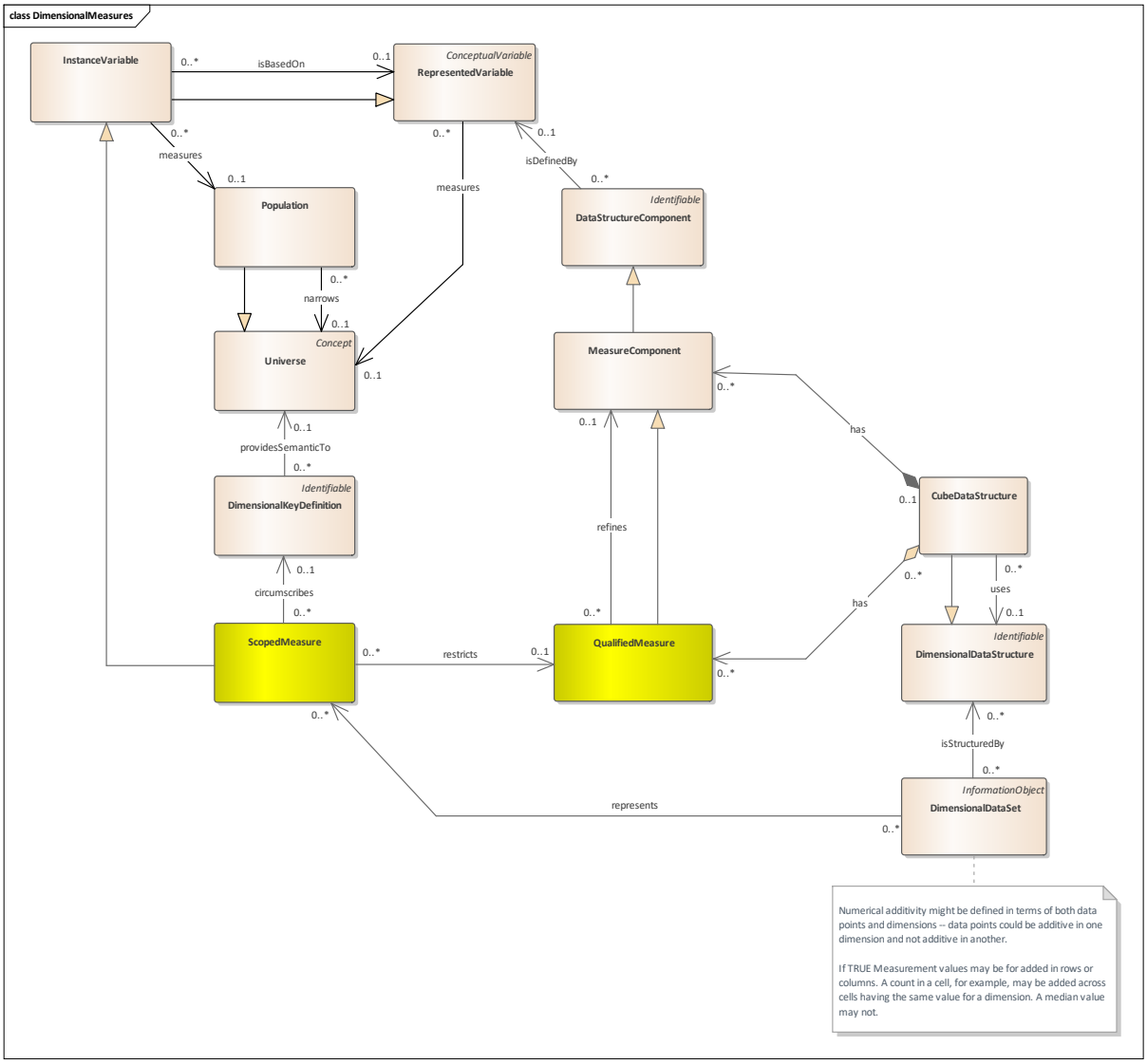


Figure 23: DimensionalMeasures

Each cell in the cube's DimensionalDataStructure can have not only measures associated with it, but also attributes (see diagram below).

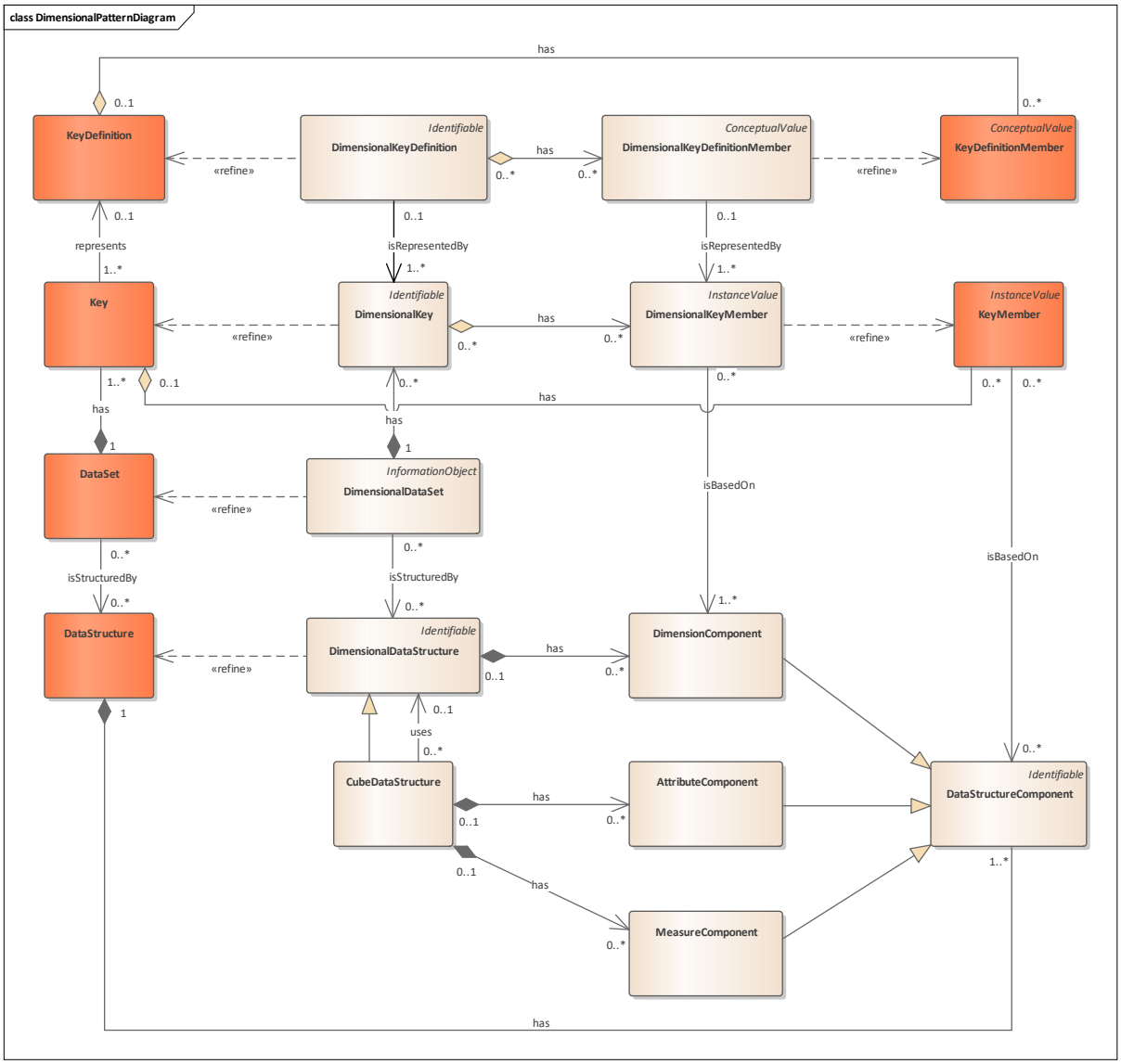


Figure 24: DimensionalPatternDiagram

The table below shows a long representation of a cube with three DimensionalComponents, one MeasureComponent, and two AttributeComponents. The attributes in this case indicate revised data in the cells of the cube, identified by vintage, and with an indication of what revision process took place.

DimensionalComponents *QualifiedMeasure* *AttributeComponents*

RefArea	Sex	TimePeriod	AverageLongevity	Vintage	Vintage Reason	Revision process
Newport	Female	2004 - 2006	80.7	**Aug-09**	additional data	recalculate mean
Newport	Female	2005 - 2007	80.9	**Aug-09**	additional data	recalculate mean
Newport	Female	2006 - 2007	81.5	**Aug-09**	no change	none
Newport	Male	2004 - 2006	76.7	**Aug-09**	additional data	recalculate mean
Newport	Male	2005 - 2007	77.1	**Aug-09**	additional data	recalculate mean
Newport	Male	2006 - 2007	77.0	**Aug-09**	additional data	recalculate mean
Cardiff	Male	2004 - 2006	78,7	**Aug-09**	additional data	recalculate mean
Etc.						

Figure 25: Attribute components for a cube – long representation

In the diagram below, we can see that multiple Datums can exist for those cases where there are revisions: these would share a Key but would be distinguished by the vintage property associated with each RevisableDatum.

While such revisions can be handled in other ways using this model (a time stamp associated with the observation – observation period - functioning as a dimension, for example) many systems use the approach modeled here, and do not manage revisions as part of the dimensionality of their data. The requirement is that two values with identical Keys be distinguishable – this model includes RevisableDatum to support those systems which require it.

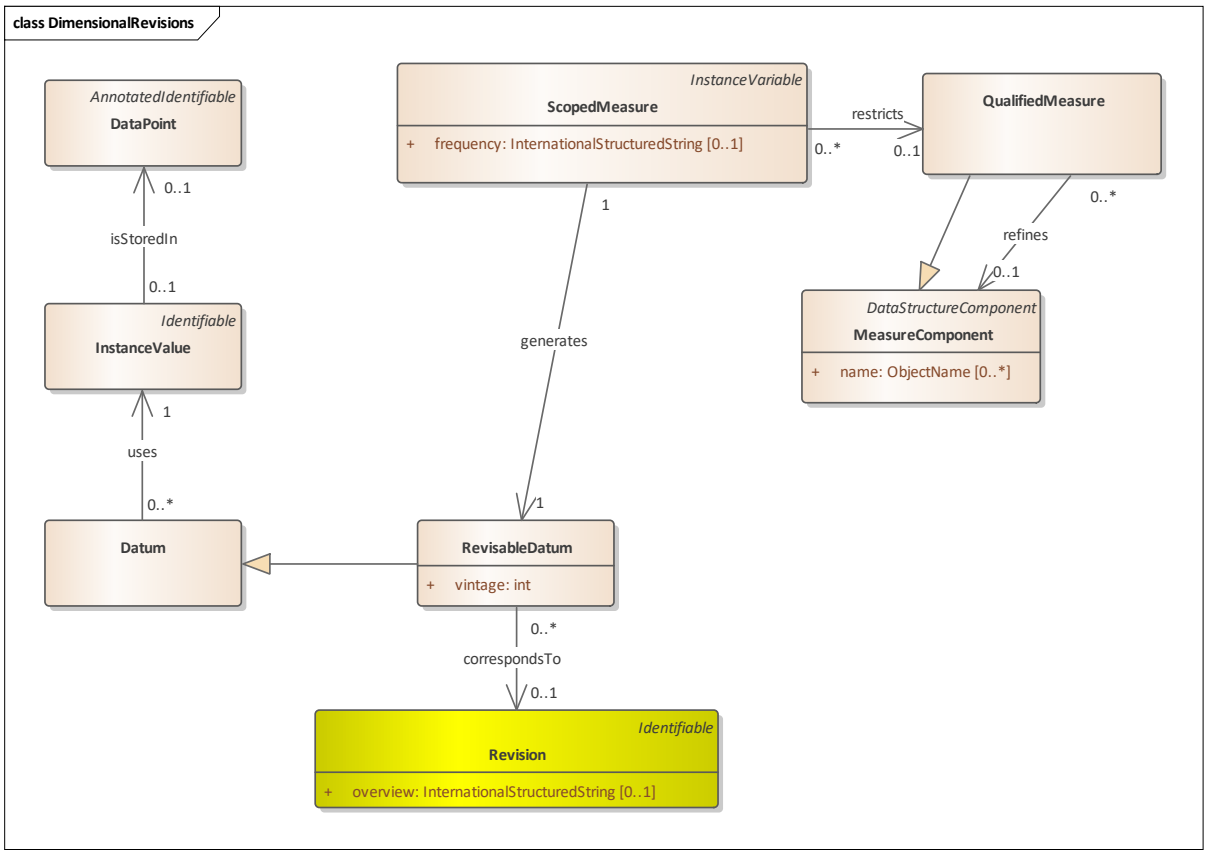


Figure 26: DimensionalRevisions

H. Key-Value Format

1. Example

A Key-Value store represents a repository holding data as a set of pairs, a key – the InstanceKey - and its associated value, a DataPoint. The DDI-CDI model is shown in figure 14. A key is a unique value that allows look-up of its linked value. The DDI-CDI model includes a KeyValueDataStore which contains the key-value pairs.

There are many possible ways to compose keys. The KeyValueDataStore may be divided into contexts, within which all of the subordinate keys are unique. The subject of the data – either a Unit or Population – can be contained as a component of the key. When this is a population, this portion of the key may itself be composed of the dimensional identifiers of the population, as for multi-dimensional data. Time may serve as a component of the key. Reference values may be used, as may variables. If needed, a "synthetic" component may be used, which holds no meaning but is unique within the context of the key.

In the example below the data are stored as key-value pairs. The Key column contains InstanceKey values that identify the associated DataPoints. Looking at the data in Figure 27, the value "3.3.1992" could be associated with a key "Marie-Born" combining the unit identifier ("Marie") and the variable

name (“Born”). The date 3.3.1932, for example is described by the InstanceKey “Marie-Born”. The cell containing 3.3.1932 is the DataPoint identified by the Key. This table, if combined with other data with keys composed in different ways, add a conxtext – a Contextual component – to the key to distinguish between the different ways in which data are being composed within the repository.

The KeyValue structure can be used for data in data lakes, No SQL systems, and other forms of big data.

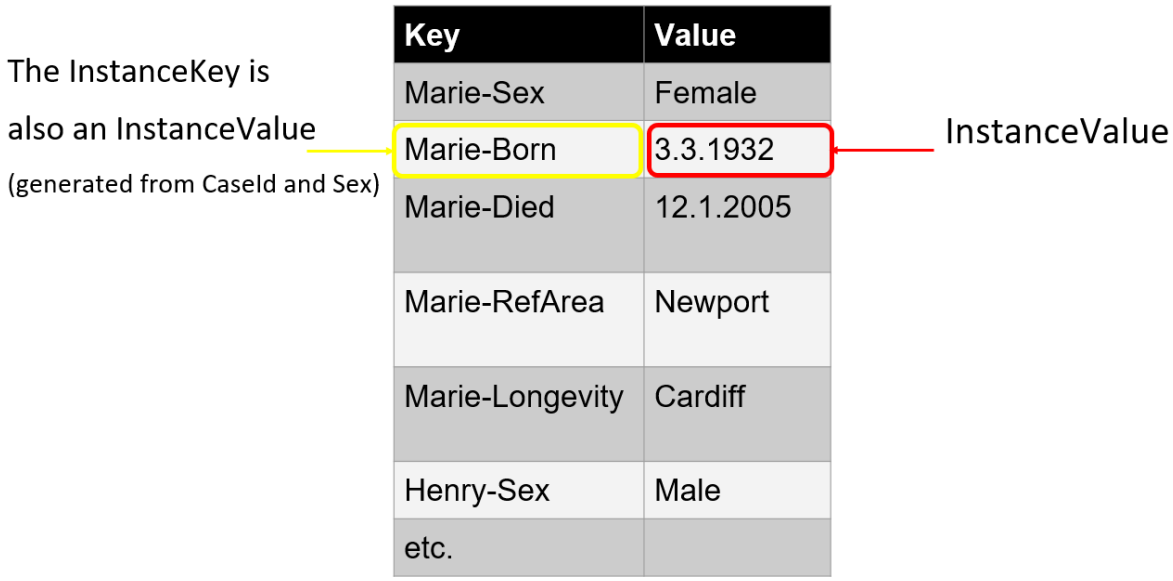


Figure 27: Key-Value Store

2. Discussion of structure and diagrams – Key-Value

At its heart the Key-Value model is simple. A key identifies a value, and a set of these are help in a KeyValueDataStore. The key is represented in DDI – CDI as an InstanceKey, the value as a DataPoint. The structure of the KeyValueDataStore is known from the KeyValueStructure with which it is associated.

It is possible to have more than one scheme for the composition of keys, by including in each a component which represents that scheme – or “context” – within which the key is unique.

The diagram below gives an overview of the relevant classes in DDI – CDI:

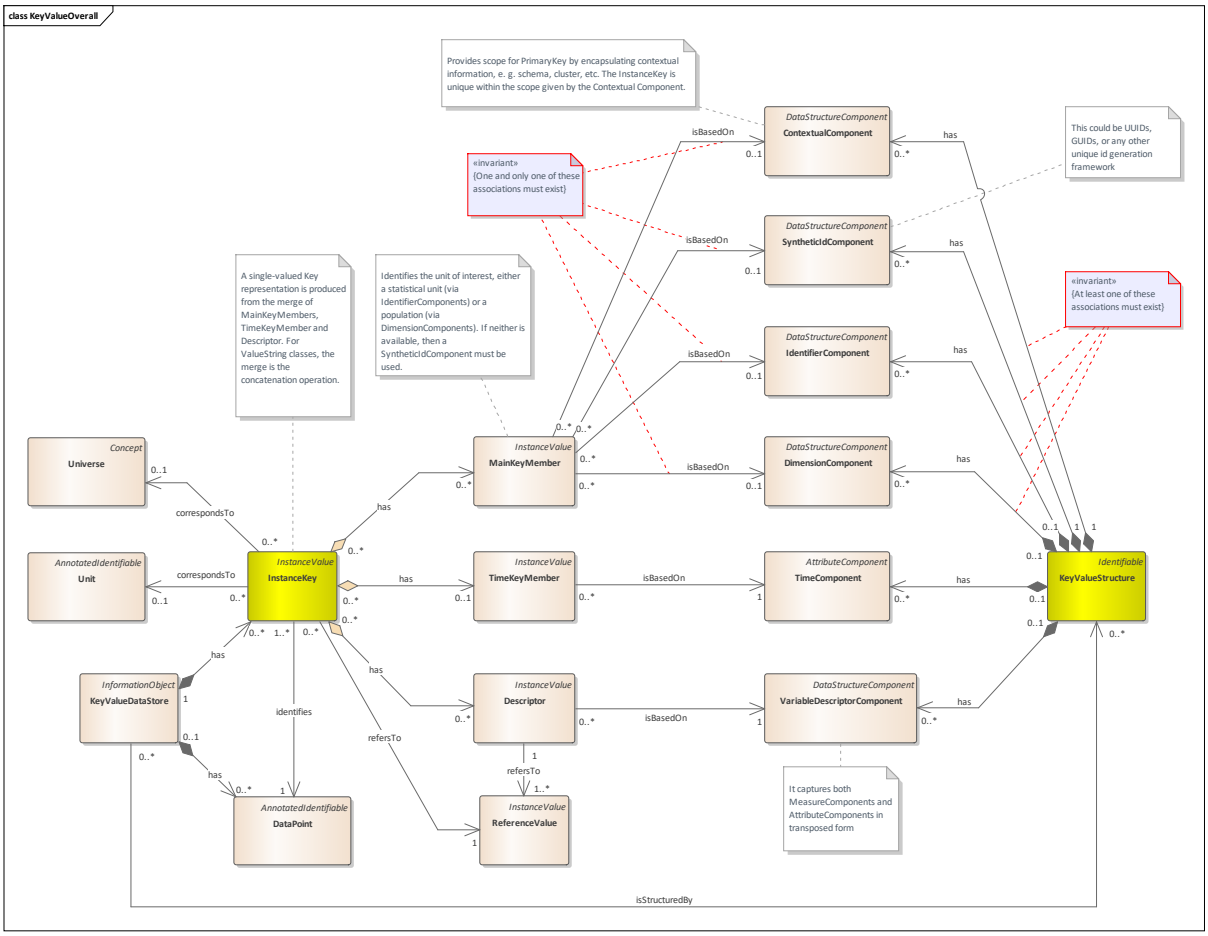


Figure 28: KeyValue overall diagram

InstanceKeys may be composed of a variety of different members: MainKeyMember, TimeKeyMember, and Descriptor are all used. These members are in turn composed of different StructureComponents according to rules which guarantee their uniqueness.

The members which are used to compose an InstanceKey are shiown in the diagram below:

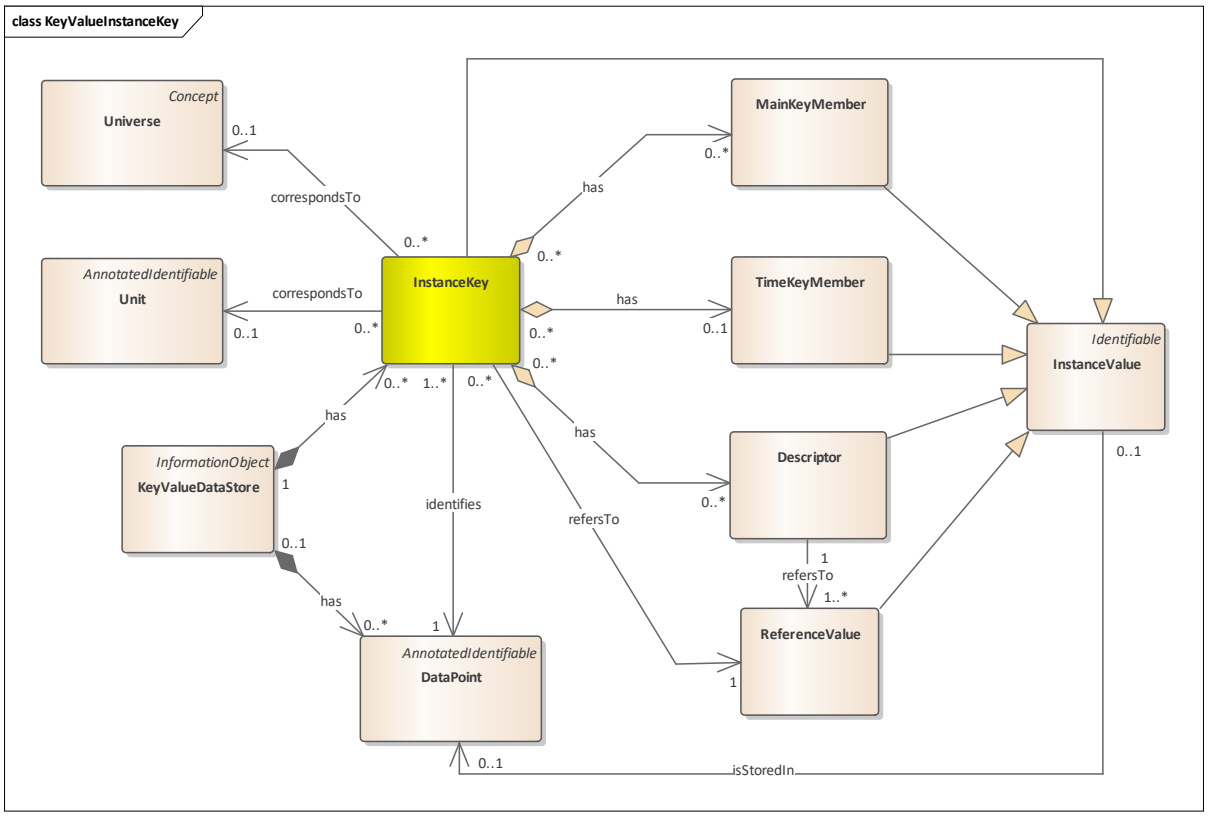


Figure 29: KeyValueInstanceKey

The MainKeyMember is the most complex one. In the simplest case, it may be composed of a SyntheticIdComponent, which might be a GUID or similar identifier which is guaranteed to be unique, but serves no other purpose. IdentifierComponents may be used to provide unitary values which identify the Units of the value (that is, their subject). Similarly, Units and Populations may be identified using DimensionComponents, providing a compound key structure like that found for multi-dimensional data. If more than one approach to composing keys is used, each may be established as a "context", and this can be added to the keys using the ContextualComponent.

TimeKeyMembers are made up of TimeComponents, which may be anything with a temporal association (this can be an enumerated value such as "Valid", a timestamp, or any other time-related value.)

Descriptors use the VariableDescriptorComponent, which brings together AttributeComponents and MeasureComponents (as for the Long Data structure). Descriptors are associated with a ReferenceValue – that is, the value held as an instance of the component being used to compose the key. (In our example, the variable "Born" could be a column in a Wide table, or a value in a Long table in the VariableDescriptor column. For Key-Value data, it is used as a Member in composing the Key.)

The StructureComponents making up the various Members may be seen in the diagram below:

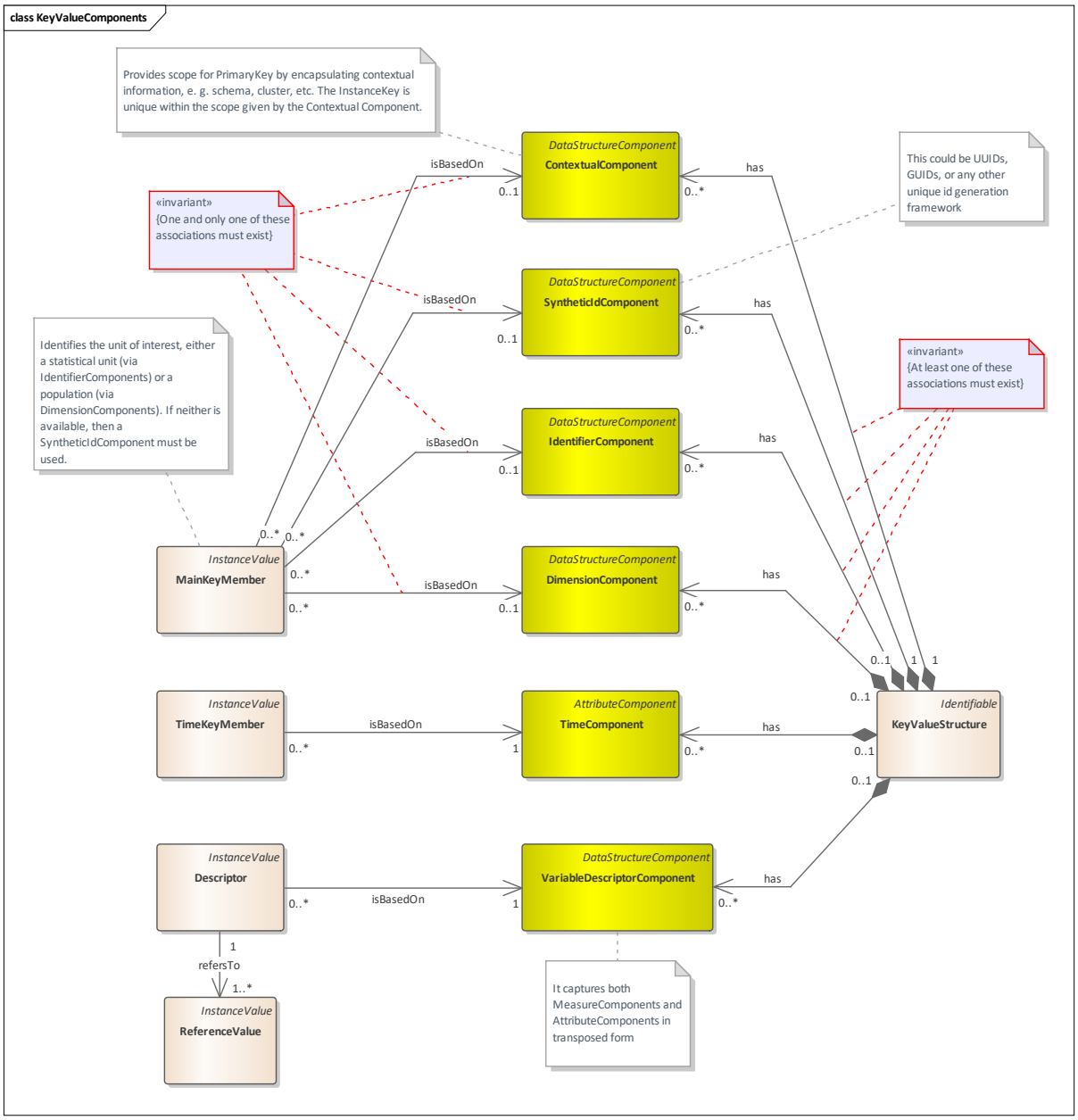


Figure 30: KeyValueComponents

A Key has a structure consisting of all of these components.

H. Physical Data Set (Wide Format)

The PhysicalDataSet diagram below shows the relationship of the PhysicalDataSet to other classes. A PhysicalDataset is a set of record segments (PhysicalRecordSegments). In older data files it was common to have a record (a row of a table) that was represented as a sequence of shorter records (e.g. strings) due to constraints imposed by the physical media. A record, for example, of 150 characters required two 80 column cards. A property of the PhysicalDataSet signifies the number of segments per record.

The order of the PhysicalRecordSegments is specified by a set of PhysicalRecordSegmentPositions, each of which having an integer describing the position of the segment in the dataset.

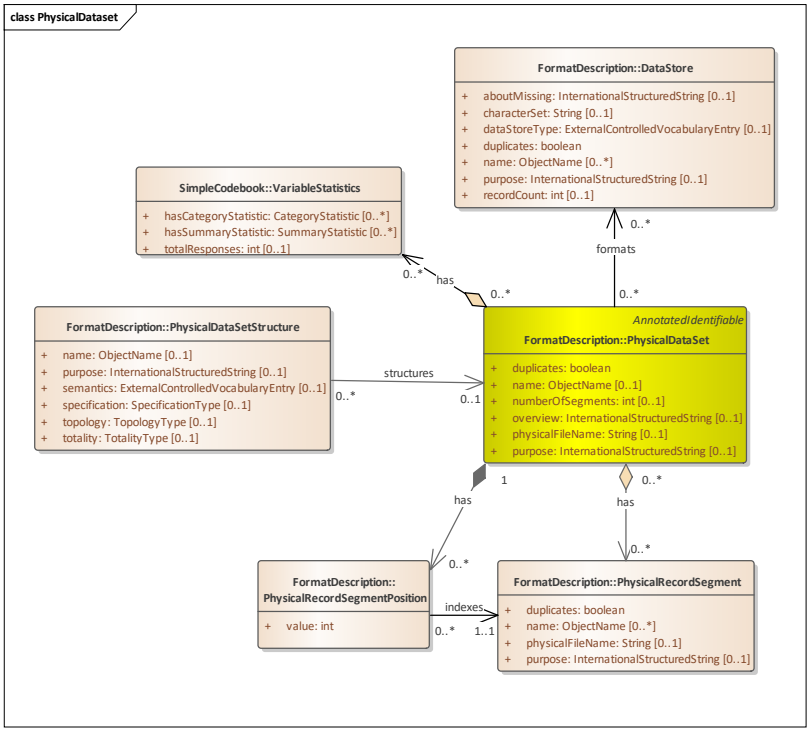


Figure 31: PhysicalDataSet overall diagram

The PhysicalRecordSegment is composed of DataPoints. A DataPoint contains an InstanceValue. In a text file the InstanceValue would be a substring of the string comprising the PhysicalRecordSegment. In a binary file it would be a sequence of bits within a larger sequence of bits. A DataPoint is described conceptually by an InstanceVariable. It is identified and set into context by a Key. The example below, for a traditional rectangular table, uses a WideKey.

The DataPoint is also described by a ValueMapping. For a string representation this contains information like the separator used for the decimal part of a number (defaultDecimalSeparator), or the maximum length of the string (maximumLength), etc.

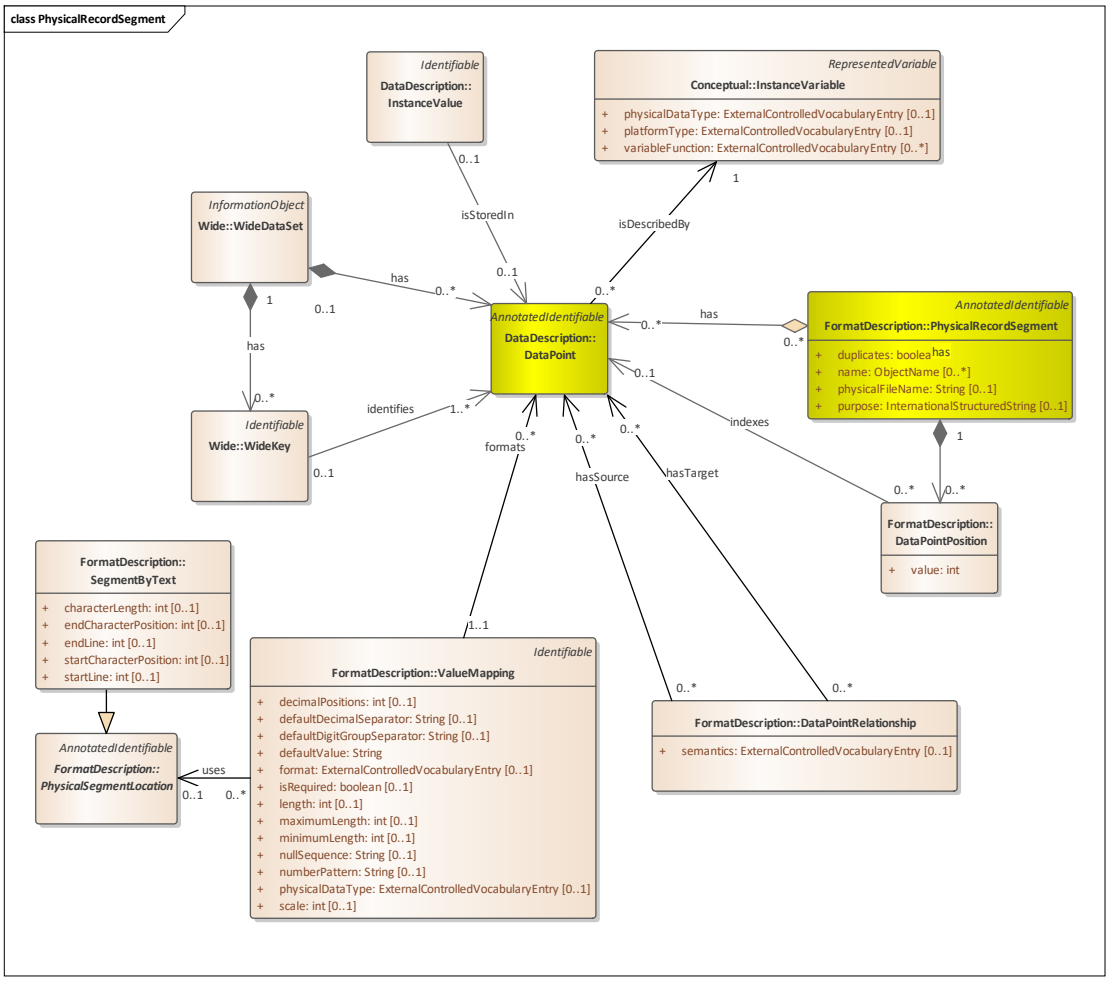


Figure 32: PhysicalRecordSegment diagram

In a text file the InstanceValue in a DataPoint is a substring of the PhysicalRecordSegment string itself. In a delimited file like a CSV file, the separation of those sequential substrings is indicated by delimiters. The PhysicalSegmentLayout contains information about those delimiters, the encoding of the record segment, whether text values are enclosed in quotes, etc..

For a fixed width file the ValueMapping can point to a SegmentByText object that contains information like the starting position (startCharacterPosition) and ending position (endCharacterPosition) of the substring within the segment. There is a parent class, PhysicalSegmentLocation, that will allow for description of data location in other types of media than text files. In a binary file this might be starting byte number and ending byte number. A video clip within a larger video file might be described by a start time and end time or by start and end frame number.

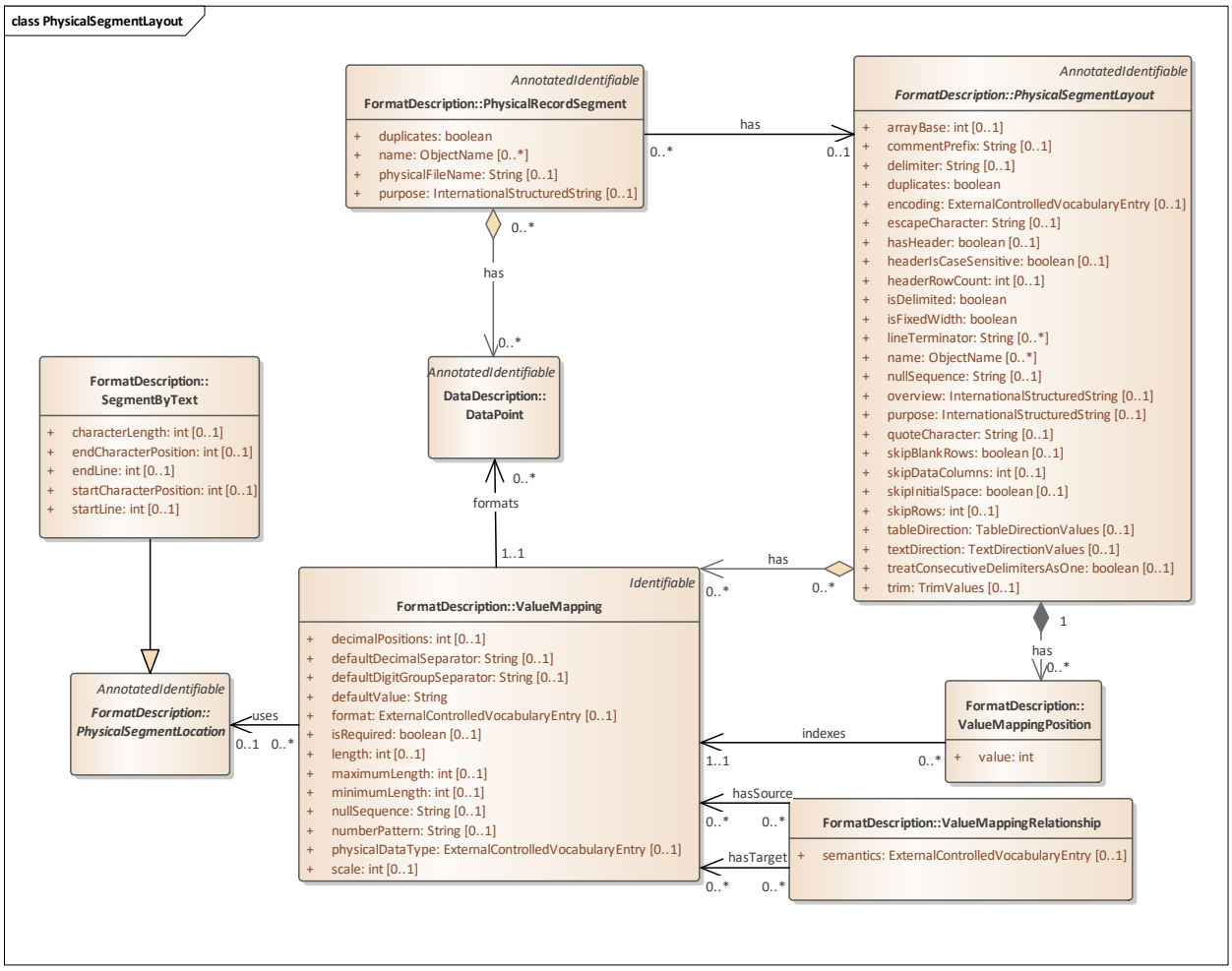


Figure 33: PhysicalSegmentLayout diagram

I. Transformations between Formats/Examples

1. Wide and Long: Correspondence between Unit record data and data in a Long format

Figure 34 below shows the mapping between the Wide Unit record format and the Long format. We see that all combinations of variables and values for each unit record identifier are retained. Each value in the record for Marie now has its own row, with a second value – the Unit Variable – telling us what the value is (the column in the Wide table). The cell value is the InstanceValue.

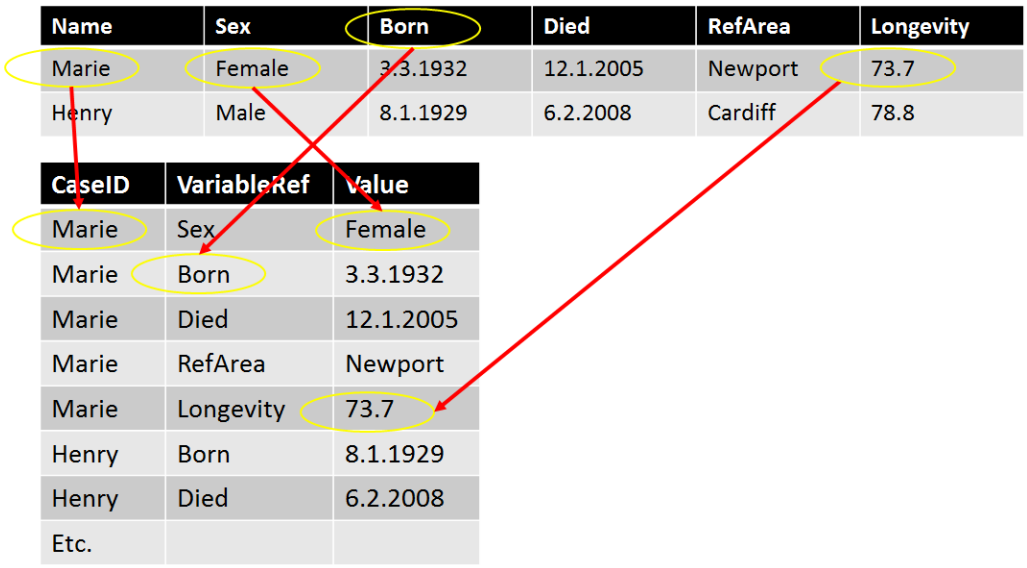


Figure 34: Transformations from Wide to Long format.

The VariableDescriptorComponent allows for the tracking of Datums between traditional wide layouts like the unit record format and Long layouts as shown in the Figure 34 example. All of the popular data analysis platforms have procedures like the R and Stata "reshape" function, or SAS PROC Transpose, that transform data tables back and forth between the two layouts. DDI – CDI provides a way to record this metadata which is not typically supported by non-proprietary formats.

Some types of data, like event data, typically employ Long layouts for the flexibility of adding measures and for the ability to represent sparse structures economically. Documenting these layouts with earlier versions of DDI (e.g., DDI Codebook, DDI Lifecycle) has been problematic. Columns like "Value" in the Long layout example cannot be described as a traditional variable with a single value domain. They are instead a set of Datums having different conceptual domains and representations.

The ValueMapping attached to the DataPoint allows for description of the physical representation of the generic representations in the Value column. That column as a whole must have a common representation, like a text string or bit string, that is capable of representing all of the value types for the set of underlying InstanceVariable.

2. Wide and dimensional: Unit record data tabulated into an aggregate data Cube

Unit record data can be tabulated into cubes (aggregate/dimensional data). Data from the individual units contribute to the aggregates of a cube. We see that 'Mary', 'Henry' and the others contribute to the aggregate statistics of the cube. The appropriate Unit record datum are averaged, producing the datum for the cube cell. In the cube below Marie contributes to two different cells due to overlapping time periods, while Henry only contributes to one cell.

PersonID	Sex	Born	Died	RefArea	Longevity
Marie	Female	3.3.1932	12.1.2005	Newport	73.7
Henry	Male	8.1.1929	6.2.2008	Cardiff	78.8
Etc.					

RefArea	Sex	TimePeriod	AverageLongevity
Newport	Male	2004-2006	76.7
Newport	Male	2005-2007	77.1
Newport	Male	2006-2008	77.0
Newport	Female	2004-2006	80.7
Newport	Female	2005-2007	80.9
Newport	Female	2006-2008	81.5
Cardiff	Male	2004-2006	78.7
Cardiff	Male	2005-2007	78.6
Cardiff	Male	2006-2008	78.7
Etc.			

Figure 35: Unit record data transformed to cube data

When computing a cube from the unit record data the value domains of some of the variables listed as measures above will correspond to dimensions of the cube. The categories of Sex, for example define the Sex dimension in the cube example. A computation on Died above would produce the time categories for the cube. The combination of dimension values for each unit (person here) would determine which set of units would contribute to the computation of the measure (Longevity here).

The SQL query that follows computes the cube data from the unit data:

```
create table WalesCube as
select Sex,RefArea,
    case
     when died >= '1jan2004'd and  died <= '31Dec2006'd   then "2004-2006"
     when died >= '1jan2005'd and  died <= '31Dec2007'd   then "2005-2007"
     when died >= '1jan2006'd and  died <= '31Dec2008'd   then "2006-2008"
    else " "
    end as TimePeriod,
    mean(Longevity) as Longevity

 from WalesUnitData
 group by sex, TimePeriod, RefArea
 ;
```

The processing code used to perform aggregations can be expressed in many different forms, both standard and proprietary. The Process model of DDI – CDI is designed to work with these to connect the metadata describing the data (both pre- and post-transformation) with the relevant processing code.

3. Long and Dimensional: Dimensional data represented in a Long data format

As noted before dimensional data can be represented in a Long layout. In this case the measure corresponds to the QualifiedMeasure in the model. Its population is the whole set of observations in the cube. There could be an extra column to represent the vintage instance for the associated measure. The DDI-CDI model includes classes that can assign roles to variables. In this example the first three variables take on the role as an IdentifierComponent. The values (codes), like “Newport”, or “2005-2007”in those columns are the representations of IdentifierComponent in the model. The longevity variable has a MeasureComponent, and the revision variable is an AttributeComponent. The values (codes) like “Newport”, or “2005-2007” in those columns are the representations of the IdentifierComponents in the model.

	IdentifierComponent		*QualifiedMeasure*	
Geography	**Gender**	**Time**	**Longevity**	**Vintage**
NewPort	Male	2004-2006	76.7	**Aug-09**
NewPort	Female	2004-2006	80.7	**Aug-09**
NewPort	Male	2005-2007	77.1	**Aug-09**
NewPort	Female	2005-2007	80.9	**Aug-09**
NewPort	Male	2006-2008	77	**Aug-09**
NewPort	Female	2006-2008	81.5	**Aug-09**
Cardiff	Male	2004-2006	78.7	**Aug-09**
Cardiff	Female	2004-2006	83.3	**Aug-09**
…	…	…	…	..
Merthyr	Male	2006-2008		**Jul-09**
Merthyr	Female	2006-2008		**Jul-09**

Figure 36: Dimensional as Long

4. Key-Value and Wide: Key-Value Stores in RAIRD

The example bellow shows what a possible dataset based on the RAIRD information model might look like. (RAIRD is a project involving the compilation of data from a set of administrative registers in Norway into a resource which can be used securely through an online analysis package by researchers. The central compiled data store is similar to the example given here, but researchers perform analysis on Wide data sets derived from it. The data is a form of “event history” data, giving information about specific events and periods for the Units it describes.)

RAIRD uses a hybrid form of Long and Wide layouts in that they add StartDate and EndDate as attributes that identify a value. In Figure 37 we recognize the crosswalk from the Wide Unit record data format to Long. StartDate and EndDate variables for each value are added additionally.

The keyValue table expresses the collection of variables in a possible RAIRD data set and how they are ordered. Key values link roles to each of them.

PersonID	Sex	Born	Died	RefArea	Longevity
Marie	Female	3.3.1932	12.1.2005	Newport	73.7
Henry	Male	8.1.1929	6.2.2008	Cardiff	78.8

raird:keyValue

InstanceVariableID	IndexValue	Role
CaseID	1	Identifier
VariableReference	2	Identifier
Value	3	Measure
StartDate	4	Attribute
EndDate	5	Attribute

RAIRD DataSet

CaseID	VariableRef	Value	StartDate	EndDate
Marie	Sex	Female	3.3.1932	12.1.2005
Marie	Born	3.3.1932	3.3.1932	12.1.2005
Marie	Died	12.1.2005	12.1.2005	12.1.2005
Marie	RefArea	Newport	1.1.2003	12.1.2005
Marie	Longevity	73.7	12.1.2005	12.1.2005
Henry	Born	8.1.1929	8.1.1929	
Henry	Died	6.2.2008	6.2.2008	
Etc.				

Figure 37: Example from the RAIRD information model

Here, we see that both CaseID and VariableRef function as identifiers – taken together, they uniquely identify a record in the Long format, and indeed as the identifier for a specific measure (the Value).

5. Time Series

With time as an attribute dimension in a full cube, a time series can be seen as a slice of the cube, holding the structural identifier values constant. In the example below geography and Gender are held constant and time varies across its possible values. The vintage column is added to indicate which revision of the data is being reported.

CellDefinition			*QualifiedMeasure*	
geography	**Gender**	**time**	**longevity**	**vintage**
NewPort	Male	2004-2006	76.7	**Aug-09**
NewPort	Male	2005-2007	77.1	**Aug-09**
NewPort	Male	2006-2008	77	**Aug-09**

6. Key-Value Stores and Streams

Streaming data may involve a flexible set of measures arriving at unpredictable times. Structures that may be useful for streaming data include the tall structure (like for event data) or a key value store. With a tall structure, measure variables may be associated with identifier variables (such as a sensor identifier) and attribute variables (such as time of measurement, time of receipt, and location of measurement).

Measures may involve datatypes note currently described in DDI (images, sound recording, etc.) but envisioned as potential candidates for inclusion in future.

An example sensor observation from the W3C Semantic Sensor Network Ontology (SSN) (https://www.w3.org/TR/vocab-ssn/#iphone_barometer-sosa) is of a barometric pressure taken by an iPhone. The SSN RDF for the Observation is:

```
<Observation/346344> rdf:type sosa:Observation ;
  sosa:observedProperty <sensor/35-207306-844818-
0/BMP282/atmosphericPressure> ;
  sosa:hasFeatureOfInterest  <earthAtmosphere> ;
  sosa:madeBySensor <sensor/35-207306-844818-0/BMP282> ;
  sosa:hasSimpleResult "1021.45 hPa"^^cdt:ucum ;
  sosa:resultTime "2017-06-06T12:36:12Z"^^xsd:dateTime .
```

A tall representation for the data might look like this, where the value atmosphericPressurehPa is a code that points to a variable that links to the Concept "`earthAtmosphere`" in units of hectoPascal (hPa).

SensorID	Property	Time	ResultingValue
sensor/35-207306-844818-0/BMP282	atmosphericPressurehPa	2017-06-06T12:36:12Z	1021.45

Figure 38: Sensor reading in Tall format

A Key-Value representation might look like this. The SensorID and Property are concatenated into a single Key. The Key could be decomposed into the SensorID and Property components as needed.

Key	Time	ResultingValue
sensor/35-207306-844818-0/BMP282/atmosphericPressure	2017-06-06T12:36:12Z	1021.45

Figure 39: Sensor reading in Key-Vaölue format

IV. The Process Model

A. Introduction

The D - CDI Process model is a generic process model able to describe retrospectively a succession of activities. These activities may be a set of business processes described at a conceptual level and/or a set of concrete steps (and their steps, ad infinitum) that take InformationObjects as parameters. Additionally, these activities may be a succession of questions in a questionnaire. InformationObjects may include, data, structured metadata, and computer programs.

Several forms of "succession" can be described. They fall into two categories – deterministic and non-deterministic. Deterministic succession may be parallel or sequential. Non-deterministic succession may be temporally ordered using algebras like Allen's interval algebra. Alternatively, non-deterministic succession may be governed by inference engines that form the basis for rule-based systems.

Generally speaking, each type of succession is supported by a set of control constructs. Together the control constructs form a plan or program that orchestrates a workflow. Depending on the control constructs, there are a myriad of workflow patterns.

1. Relation to other standards

There are several models currently in use which provide a strong basis for the DDI - CDI Process model. PROV-O is perhaps the best-known of these, giving us a basic set of classes describing *Activities* (the things which are done), *Agents* (the people and organizations which do things), and *Entities* (the resources which are operated on/with and produced). This is an extremely general model, and one which was designed to be made specific for use in specific applications.

Recently, PROV-O has been extended by ProvONE. ProvONE makes PROV-O data- and computer-program-specific. In PROV-O, entities didn't distinguish data at different level of specificityThe PROV-O Plan entity lacked the specificity to describe the structure of computer programs and the specific successions of activities (workflows) that programs create.

Here is the ProvONE Conceptual Model:

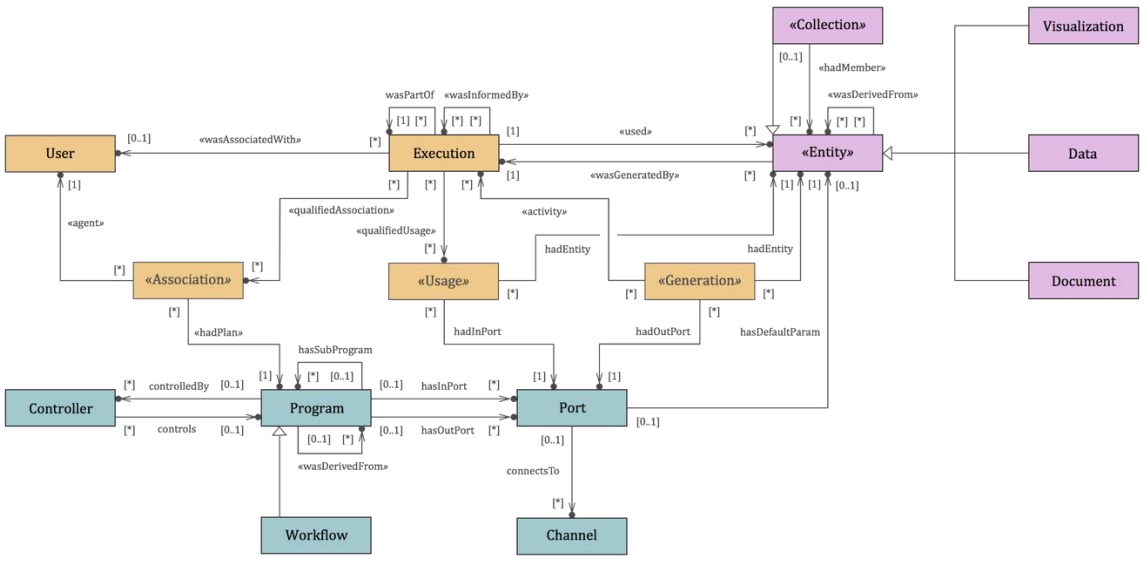


Figure 40: The PROVOne model

DDI - CDI process descriptions can be understood as extensions of PROV-O and ProvONE.

These extensions mostly take the form of ControlConstructs which DDI - CDI has borrowed from other products in the family of DDI specifications, notably DDI Lifecycle. DDI Lifecycle process components borrowed heavily from OWL-S, as shown below. (Notably, the Control Construct is a central feature of how DDI Lifecycle describes questionnaire flows.)

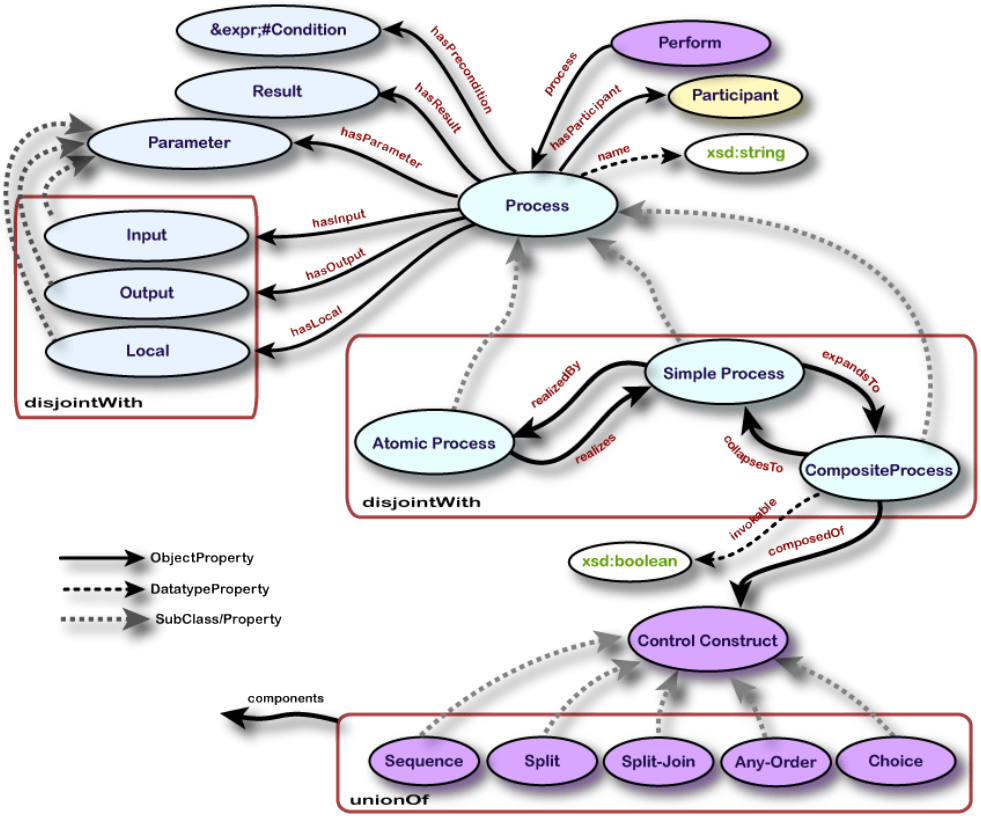


Figure 41: Control constructs mind map

2. Aspects covered by the DDI - CDI Process model

Currently “prospective provenance” and “process provenance” are not in scope. In prospective provenance plans and programs have a hand in guiding execution. Process provenance is about workflow evolution over time. Workflow evolution is integral to machine learning experiments which might evaluate a succession of workflows. (Workflow evolution may be addressed by DDI – CDI in future.)

For now, the focus is “retrospective provenance” or, again, “data lineage”. When data lineage enumerates a set of beginning and intermediate on-ramps in a workflow, it is *backward data lineage*. When data lineage enumerates a set of off ramps for InformationObjects that have entered the workflow upstream, this is *forward data lineage*. The DDI - CDI Process model aims to be able to describe both backward and forward data lineage.

B. Process Model Conceptual Model Overview

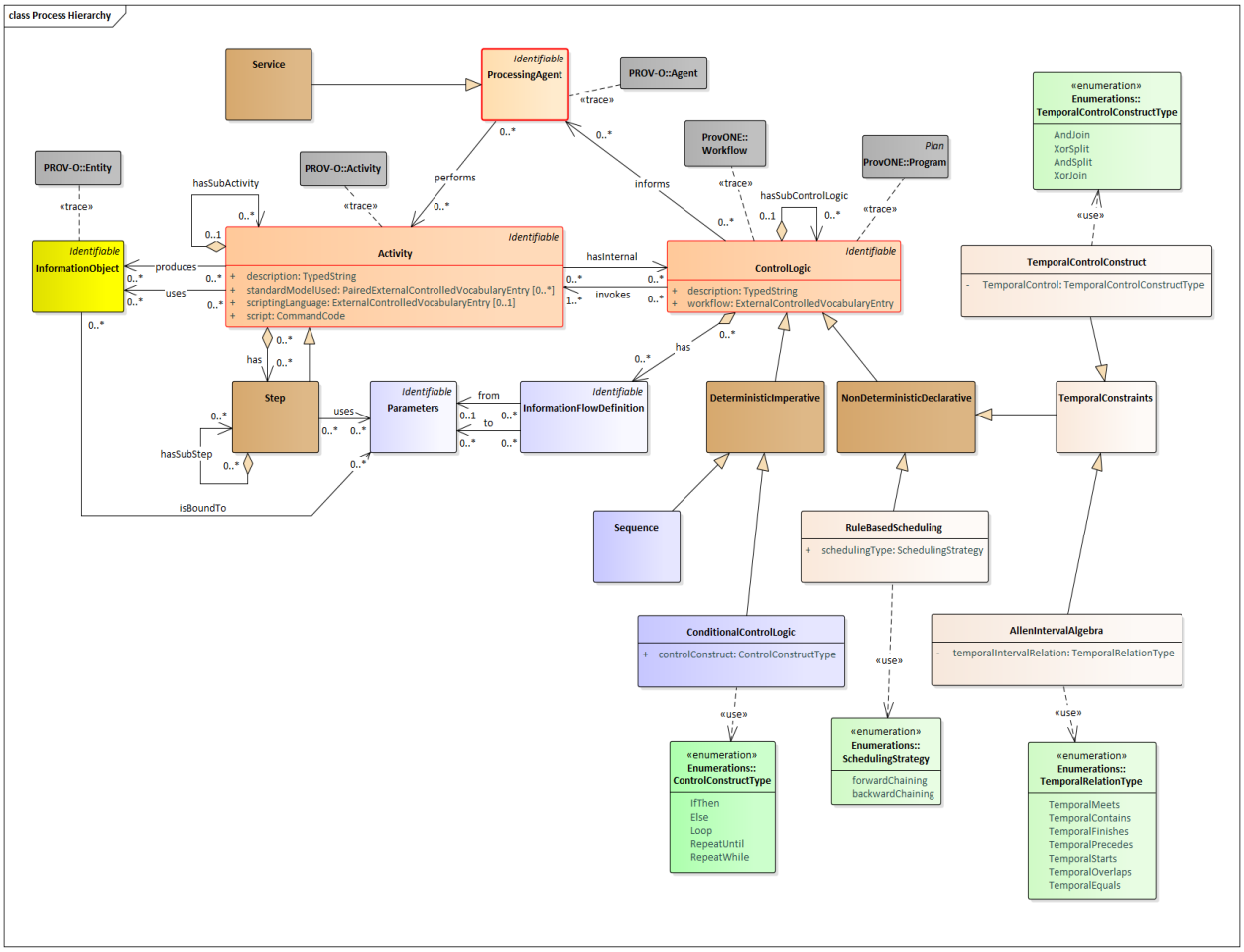


Figure 42: DDI - CDI Process model overview

In the DDI - CDI Process model ControlLogic invokes Activities. Activities may take In and Out Parameters. InformationObjects are bound to these Parameters. An InformationFlowDefinition connects the Out Parameter(s) of one Activity with the In Parameter(s) of another Activity. These connections, in the aggregate, create Workflows.

Note that an InformationObject may be data, structured metadata, or a program.

C. Process Model Conceptual Model Detail

Note that detailed documentation at for Data Description model in DDI – CDI can be found in this package in the folder: `\DDI-CDI Public Review 1\2 Model\Field-Level Doc\`.

1. ControlLogic

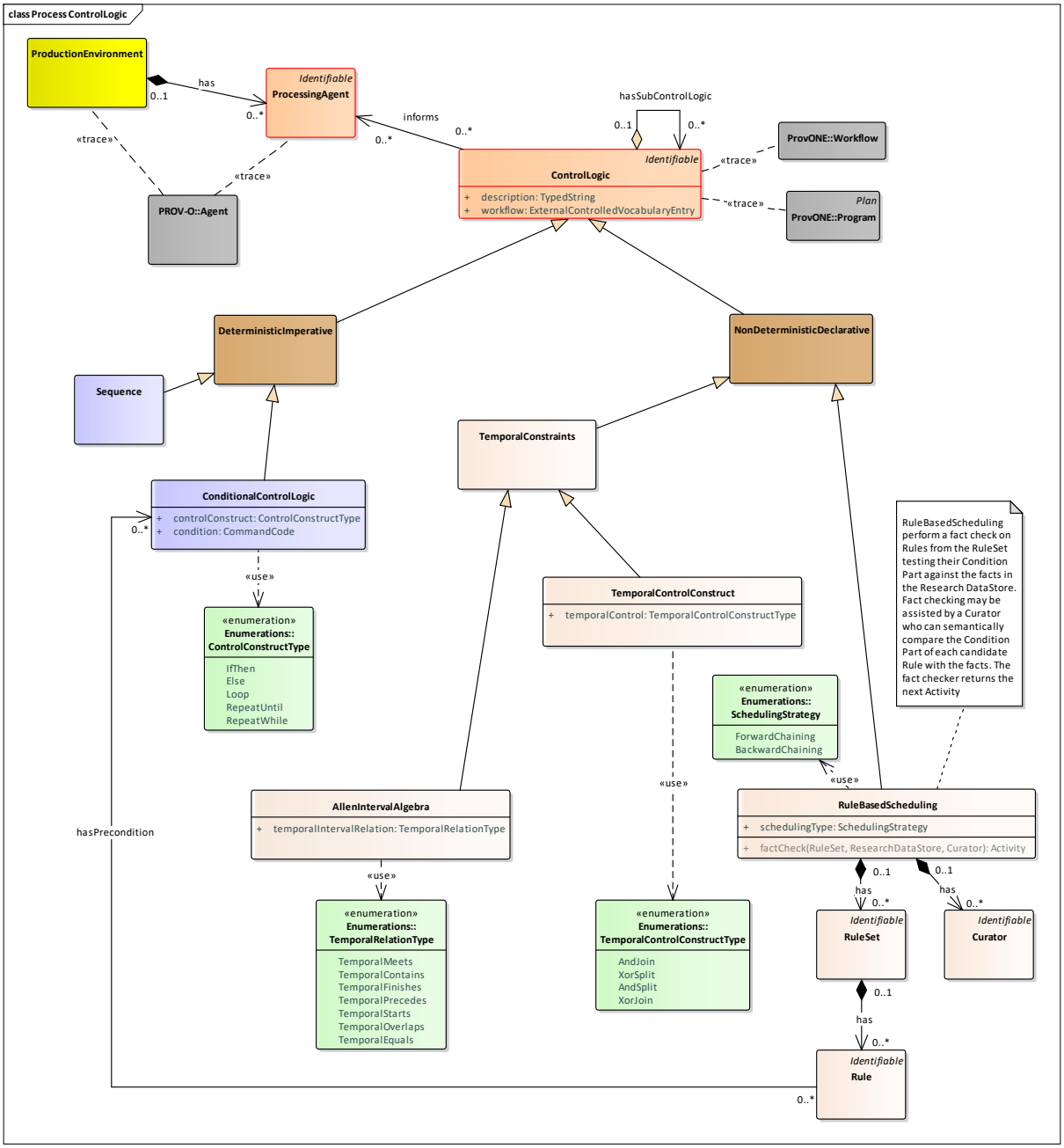


Figure 43: Process model control logic

Deterministic ControlLogic consists of Sequences and ConditionalControlLogic. Sequences may contain Sequences and ConditionalControlLogic. ConditionalControlLogic comes in several types or flavors including *If Then*, *Else*, etc. ConditionalControlLogic also includes logical expressions that evaluate to *true* or *false*. Finally, ConditionalControlLogic may contain Sequences.

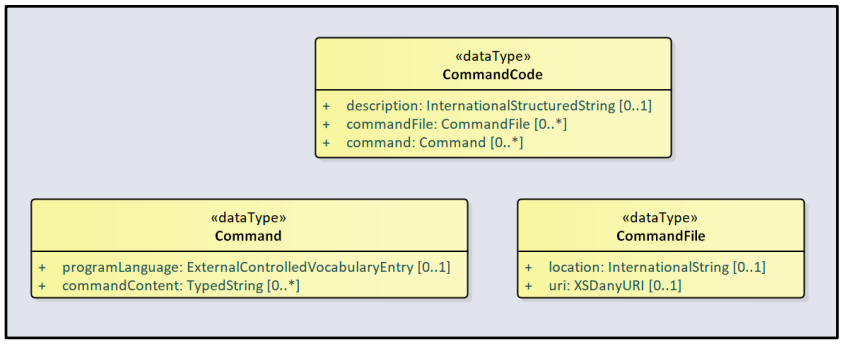
NonDeterministic ControlLogic has two subtypes – TemporalConstraints and RuleBasedScheduling. TemporalConstraints in turn has two subtypes – AllenIntervalAlgebra and TemporalControlConstruct. Both AllenIntervalAlgebra and TemporalControlConstruct use enumerations to qualify their type further. Note that AllenIntervalAlgebra is a calculus for temporal reasoning useful in describing complex pairwise temporal relationships across a group of Activities. TemporalControlConstructs, on the other hand, are useful in describing parallel processing.

RuleBasedScheduling takes a RuleSet and InformationObjects as input and produces InformationObjects as output. RuleBasedScheduling may employ the assistance of one or more domain-specific curators to match the Rule's conditions with real world facts or the current state of InformationObjects in the ResearchDataStore.

2. C2Metadata Support

DDI - CDI supports the work of the C2Metadata project. Both Activities and the Steps an Activity might contain in the Process Model host a Script made up of Commands. Each Command consists of programming language (specified in a codelist) and the Command content. Each Command may have multiple programming language / Command content pairs, which would be deemed equivalent (i.e., an executable STATA syntax example and its human-readable equivalent in SDTL – see the C2Metadata link above) .

C2Metadata takes a Script and its Command as input and produces additional Command content as output. The input is the original programming language and its command content. The output is documentation of the input. C2Metadata output may be in several languages including SDTL – the Structured Data Transformation Language.



Alongside SDTL, C2Metadata produces documentation of Commands in natural language and DDI Lifecycle.

3. Workflow

ControlLogic (the "program") in the DDI Process specification and all the deterministic and non-deterministic logic that inherit from ControlLogic have a *workflow* attribute. *workflow* is typed as ExternalControlledVocabulary (a codelist). In fact, the Workflow Patterns Initiative (WPI) has created a compendium of 40+ WorkflowPatterns noting the motivation, context, issues, issue solutions and supporting products and platforms associated with each one. In the Examples Document there are many examples in which the ExternalControlledVocabulary is the WPI where the WorkflowPattern in play in the ControlLogic is one of the 40+ patterns that the WPI describes.

```
<TemporalControlConstruct>
    <Agency>INDEPTH-Karonga-HDSS</Agency>
    <Id>Act-IntegrateData-1-TCC-AndSplit-1</Id>
    <Version>Pentaho-01</Version>
    <Workflow>
        <ControlledVocabularyAgencyName>Workflow Patterns Initiative</ControlledVocabularyAgencyName>
        <ControlledVocabularyID>WPI_Pattern_02</ControlledVocabularyID>
        <ControlledVocabularyName>Parallel Split</ControlledVocabularyName>
        <Content>
            The divergence of a branch into two or more parallel branches each of which execute concurrently.
        </Content>
        <Uri>http://www.workflowpatterns.com/patterns/control/basic/wcp2.php</Uri>
    </Workflow>
    <TemporalControl>AndSplit</TemporalControl>
</TemporalControlConstruct>
```

Several example WorkflowPatterns taken from the Workflow Patterns Initiative can be found below.

D. Illustrative WDI Workflow Patterns

Pattern 1 (Sequence)

FLASH animation of Sequence pattern

Description

A task in a process in enabled after the completion of a preceding task in the same process.

Synonyms

Sequential routing, serial routing.

Examples

The *verify-account* task executes after the credit card details have been captured.

The *codacil-signature* task follows the *contract-signature* task.

A receipt is printed after the train ticket is issued.

Motivation

The *Sequence* pattern serves as the fundamental building block for processes. It is used to construct a series of consecutive tasks which execute in turn one after the other. Two tasks form part of a *Sequence* if there is a control-flow edge from one of them to the next which has no guards or conditions associated with it.

Overview

Figure 1 illustrates the *Sequence* pattern using CP-nets.

Figure 1: Sequence pattern

Context

There is one context condition associated with this pattern: an instance of the *Sequence* pattern cannot be started again until it has completed execution of the preceding thread of control (i.e. all places such as p1 in the *Sequence* must be safe).

Here is the flash animation of that pattern.

Pattern 5 (Simple Merge)

FLASH animation of Simple Merge pattern

Description

The convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.

Synonyms

XOR-join, exclusive OR-join, asynchronous join, merge.

Examples

At the conclusion of either the *bobcat-excavation* or the *D9-excavation* tasks, an estimate of the amount of earth moved is made for billing purposes.

After the *case-payment* or *provide-credit* tasks, initiate the *product-receipt* task.

Motivation

The *Simple Merge* pattern provides a means of merging two or more distinct branches without synchronizing them. As such, this presents the opportunity to simplify a process model by removing the need to explicitly replicate a sequence of tasks that is common to two or more branches. Instead, these branches can be joined with a simple merge construct and the common set of tasks need only to be depicted once in the process model.

Overview

Figure 5 illustrates the behaviour of this pattern. Immediately after either task A or B is completed, task C will be enabled. There is no consideration of synchronization.

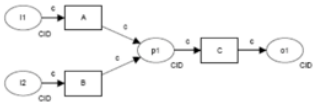


Figure 5: Simple merge pattern

Context

There is one context condition associated with the pattern: the place at which the merge occurs (i.e. place p1 fin Figure 5) is safe and can never contain more than one token.

Here is the flash animation of that pattern.

Pattern 40 (Interleaved Routing)

FLASH animation of Interleaved Routing pattern

Description

Each member of a set of tasks must be executed once. They can be executed in any order but no two tasks can be executed at the same time (i.e. no two tasks can be active for the same process instance at the same time). Once all of the tasks have completed, the next task in the process can be initiated.

Examples

The *check-oil, test-feeder, examine-main-unit* and *review-warranty* tasks all need to be undertaken as part of the machine-service process. Only one of them can be undertaken at a time, however they can be executed in any order.

Motivation

The *Interleaved Routing* pattern relaxes the partial ordering constraint that exists with the *Interleaved Parallel Routing* pattern and allows a sequence of tasks to be executed in any order.

Overview

Figure 58 illustrates the operation of this pattern. After A is completed, tasks B, C, D and E can be completed in any order. The **mutex** place ensures that only one of them can be executed at any time. After all of them have been completed, task F can be undertaken.

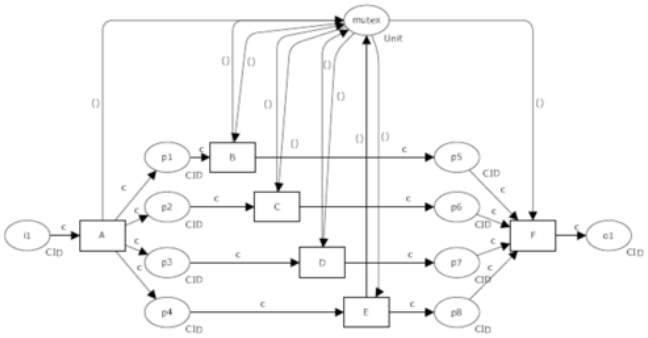


Figure 58: Interleaved routing pattern

Context

There is one consideration associated with the use of this pattern: tasks must be initiated and completed on a sequential basis, in particular it is not possible to suspend one task during its execution to work on another.

Here is the flash animation of that pattern.

A record of activities, agents and entities can be recorded in a Pathway, which provides the detailed provenance information for the data/metadata resulting from the process.

The diagram below provides a detail look at this portion of the DDI 4 State Based Process Model.

Pattern 3 (Synchronization)

FLASH animation of Synchronization pattern

Description

The convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled.

Synonyms

AND-join, rendezvous, synchronizer.

Examples

The *despatch-goods* task runs immediately after both the *check-invoice* and *produce-invoice* tasks are completed.

Cash-drawer reconciliation can only occur when the store has been closed and the credit card summary has been printed.

Motivation

Synchronization provides a means of reconverging the execution threads of two or more parallel branches. In general, these branches are created using the *Parallel Split* (AND-split) construct earlier in the process model. The thread of control is passed to the task immediately following the synchronizer once all of the incoming branches have completed.

Overview

The behaviour of the *Synchronization* pattern is illustrated by the CP-net model in Figure 3. The pattern contains an implicit AND-join, known as the *synchronizer*, which is considered to be *activated* once it receives input on one of the incoming branches (i.e. at places p1 or p2). Similarly it is considered to be *reset* (and hence can be re-enabled) once input has been received on each incoming branch and the synchronizer has fired, removing these tokens.

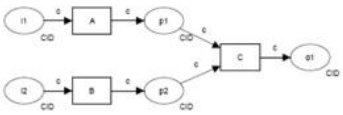


Figure 3: Synchronization pattern

Context

This pattern has the following context condition: once the *synchronizer* has been activated and has not yet been reset, it is not possible for another signal to be received on the activated branch or for multiple signals to be received on any incoming branch. In other words, all input places to the synchronizer (e.g. p1 and p2) are safe.

Here is the flash animation of that pattern.

DDI Cross Domain Integration: Architecture and Alignment with Other Standards

Contents

I. Overview

This document covers those aspects of DDI Cross Domain Integration (DDI – CDI) which deal with the internal design of the model for different purposes, and with the way in which it is expected to be used in specific implementations which are platform- and syntax-specific.

As a platform-independent model (PIM), DDI - CDI does not contain all of the information which is needed in an implementation model. It is, however, understood than implementers will wish to add additional information either to the model itself or one or more of its representations, and a framework for doing so is provided. (See the Design Notes and Modelling Approach section below for more information on what is in this review package.)

The DDI - CDI Model uses some patterns to assist in assuring that it retains internal consistency, and to make its structure consistent from the perspective of users. This is done by employing design patterns for some key features of the model.

DDI - CDI is both designed to be used with other standards and specifications – notably other DDI specifications, but also others – and is itself a user of classes from other standard models.

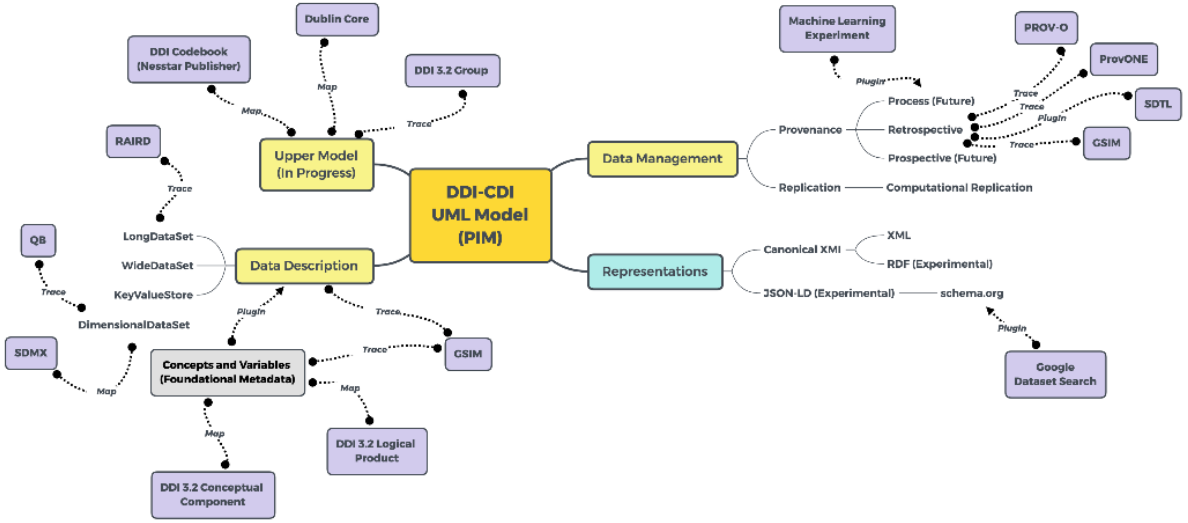
This document describes all of these features of DDI - CDI.

II. Alignment and Use of External and the DDI Family of Standards in DDI - CDI

A. Introduction

DDI – CDI is designed to support the integration of data within systems, and as such it is expected that the alignment and hand-off to other standards will be significant. In order to understand what DDI – CDI does in relation to other standards and the information they provide, we use an “Upper Model” which at this point is purely conceptual. (It is likely that in future this will become a more formal part of the DDI – CDI model, but at this point is provided for informational purposes only.) The use of some form of Upper Model may be helpful in implementing DDI – CDI, and the “Detailed Examples and Use Cases” document in this package provides an example of how this can be done.

Here is a figure that provides an overview of DDI - CDI and its relationship with other standards – both external standards like GSIM, PROV-O and schema.org together with standards in the DDI family of standards including DDI Codebook, DDI Lifecycle and C2Metadata:



Note the different types of relationships various standards have with DDI - CDI: *(UML) Trace*, *Map*, *PlugIn* and *Uses*.

B. Relationship Types between Standards

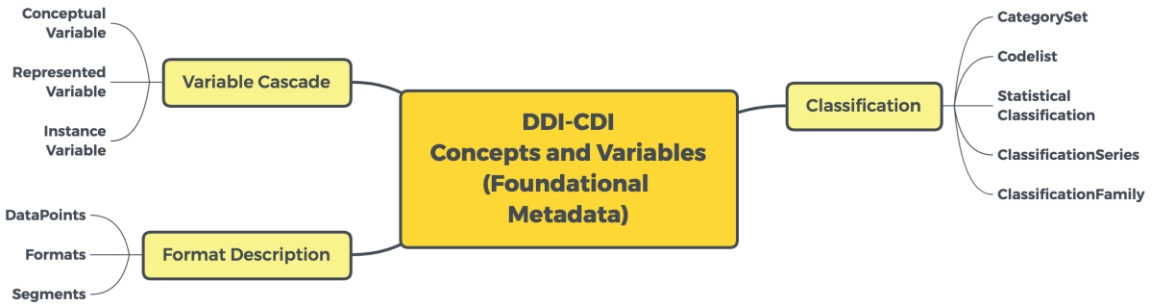
1. UML Trace

The UML Specification 2.5.1 describes the *(UML) Trace* relationship as follows:

> *"Specifies a trace relationship between model elements or sets of model elements that represent the same concept in different models. Traces are mainly used for tracking requirements and changes across models. As model changes can occur in both directions, the directionality of the dependency can often be ignored. The mapping specifies the relationship between the two, but it is rarely computable and is usually informal."*

Because trace relationships are informal, they contribute to the understanding of what is intended in the DDI - CDI model. This may not be sufficient to inform a mapping on the technical level. The trace relationship appears only in the documentary diagrams of the DDI - CDI specification, and not in the canonical model expressed in UML (including the XMI expression).

In a second figure DDI - CDI Foundational Metadata which "traces" to the DDI 3.2 (Lifecycle) Conceptual Component and the DDI 3.2 (Lifecycle) Logical Product is presented in more detail:



2. Map

In addition to (UML) Trace, there is also a *Map* relationship between a DDI - CDI component and another standard. In a Map relationship corresponding properties and/or classes are noted together with information about quality of the match using SKOS terminology: *exactMatch*, *closeMatch*, *broadMatch* and *narrowMatch*. Across a Map relationship it becomes possible to compute elements of one standard from the other, depending on the quality of the match and the direction of the relationship.

3. PlugIn

PlugIn is another type of relationship. In a PlugIn relationship between UML classes and/or their properties it is possible to plug in components from one UML model into another. PlugIn facilitates the constructions of unique profiles that are more or less in line with ISO 10000.

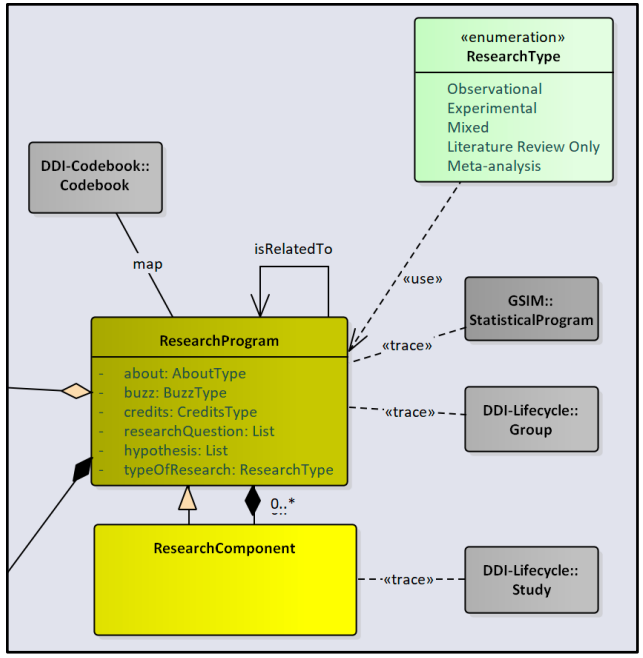
C. Alignment Specifics between DDI - CDI and other Standards in the DDI Family of Standards

1. DDI Codebook 2.5 (Nesstar Publisher)

a. Relationship with DDI - CDI: Map relationship with Upper Model

b. Description:

The DDI - CDI Upper Model includes a ResearchProgram class. It is composed of zero or more ResearchComponents that inherit from ResearchProgram. ResearchProgram includes three property groups – *about*, *buzz* and *credit*. *About* refers to the function and/or coverage of a ResearchProgram or ResearchComponent. *Buzz* refers to its social media profile – the audience, their comments, the reviews an audience might publish and so forth. Finally, *credit* follows CRediT and the Contribution Role Hierarchy and specifies the roles humans and/or programs play in the creation of a ResearchProgram and/or its ResearchComponents.



It is important to note that one or more of these property groups may be of use in other classes too. Consider a research center with a ResearchProgram that builds many research products (ResearchComponents) where some of the ResearchComponents relate to other ResearchComponent to form one or more time series and/or one or more sets of cross-sectional research products grouped by theme. The

ResearchProgram and/or its ResearchComponents may be associated with a ResearchDataStore which contains InformationObjects we might also want to *credit* and/or *buzz* and/or *about*. Indeed, this might as well be the case with data and metadata elements that compose these InformationObjects too.

Currently, in the Core Foundational Metadata there is a citation class called Annotation from which most of the foundational classes inherit. This complicates the foundational classes, one would like to say, immeasurably. As such, Annotation *does not* represent the future approach of Core with citation. Instead the Upper Model is a laboratory DDI - CDI is using to hatch the future approach. In one approach the various property groups might be implemented as a set of complex (structured) data types that other classes like ResearchProgram and ResearchComponent might specify or not as properties. In another implementation citations might become plugins to the model from other models. And so forth.

A table of these properties and their sources by property group is included in Appendix (A).

Additionally, Appendix (B) is a map between Codebook and the classes and property groups from the Upper Model.

And Appendix (C) is a poster from a map between Codebook and the Core foundational classes. This map is noteworthy in several respects. First, the map includes the *paths* one would follow through the foundational classes to get to the DDI - CDI foundational class property corresponding to a DDI 2.5 "leaf" in Codebook. Secondly, these paths are model dependent so, for example, a change in the relationship between classes in the model changes the map. And, finally, this map, using the *paths*, is *machine actionable*. This is in contrast to the map in Appendix (B) which is more conceptual.

2. DDI Lifecycle 3.2 Group Component

a. *Relationship with DDI - CDI:* Trace relationship with Upper Model

b. *Description:*

The DDI - CDI Upper Model includes a ResearchProgram class. It is composed of zero or more ResearchComponents that inherit from ResearchProgram. The DDI 3.2 Group component has a trace relationship with these two Core classes.

3. DDI Lifecycle 3.2 Conceptual Component and Logical Product Component

a. *Relationship with DDI - CDI:* Map relationship with Foundational Metadata (Concepts and Variables)

b. *Description:*

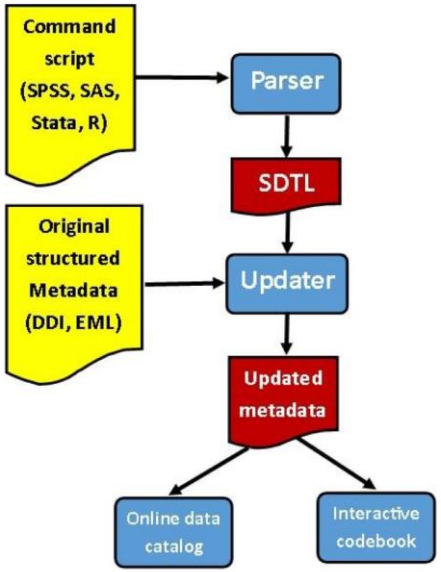
The map is a future because DDI Lifecycle 3.2 doesn't support Statistical Classification the way it is supported in DDI Lifecycle 3.3 (in evaluation) and the DDI - CDI Foundational Metadata

4. Structured Data Transformation Language

a. Relationship with DDI - CDI: Plugin relationship with Data Management Model

b. Description:

C2Metadata's Structured Data Transformation Language (SDTL) is an independent intermediate language for representing data transformation commands. Commands in four software packages (SPSS, Stata, SAS, and R) are translated into JSON schemas, which are machine actionable.



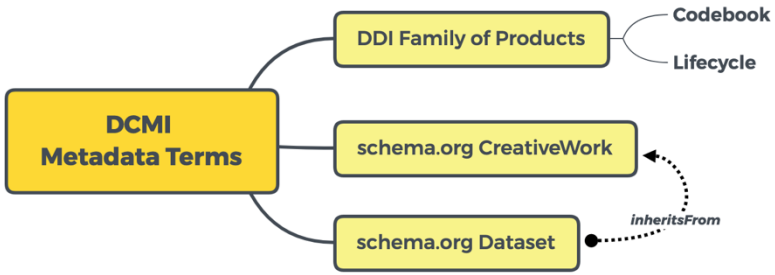
D. Alignment Specifics between DDI - CDI and External Standards

1. Dublin Core Metadata Initiative Metadata Terms

a. Relationship with DDI - CDI: Map relationship with Upper Model

b. Description:

The Dublin Core Metadata Initiative (DCMI) Metadata Terms is a superset of the Dublin Core Metadata Element Set (DCMES). Historically speaking, these vocabulary terms – the core together with its extension -- are the basis for all digital resource description. Indeed, DCMES and its DCMI Metadata Terms extension are core vocabulary terms in both DDI Codebook and DDI Lifecycle in the DDI family of products. DCMES and its DCMI Metadata Terms extension are vocabulary terms that are also at the core of schema.org CreativeWork which, in turn, provides the context in schema.org for the schema.org Dataset.



Previously, it was noted that the DDI - CDI Upper Model includes a ResearchProgram class which is composed of zero or more ResearchComponents that inherit from ResearchProgram and that ResearchProgram includes three property groups – *about*, *buzz* and *credit* – that are aligned with DDI Codebook. These same property groups are also aligned with DCMI Metadata Terms and a map between DCMI Metadata Terms and the DDI - CDI Upper Model has been included in the Appendix (C).

2. schema.org Dataset

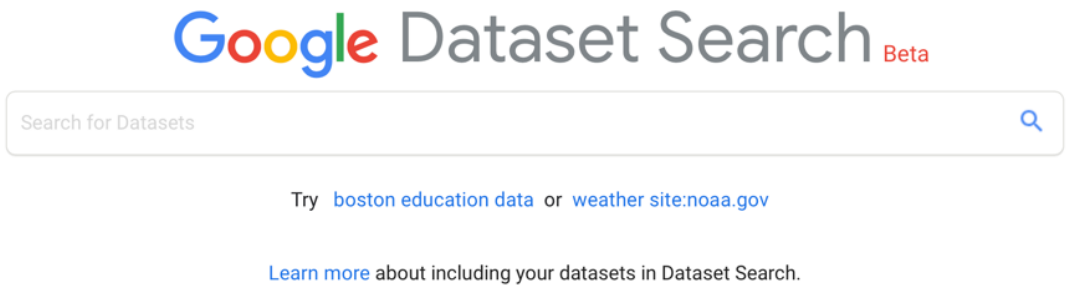
a. *Relationship with DDI - CDI:* Map relationship with Upper Model

b. *Description:*

schema.org is "a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond. In addition to people from the founding companies (Google, Microsoft, Yahoo and Yandex), there is substantial participation by the larger Web community, through public mailing lists such as public-vocabs@w3.org and through GitHub. See the releases page for more details" (from About schema.org).

The schema.org Dataset is "a body of structured information describing some topic(s) of interest" consisting of the Dataset context (CreativeWork), the Dataset variables measured each of which has a PropertyValue that includes the value, its type, any semantics associated with the type like codes and their concepts, a unit identifier, the measurementTechnique together with an extension mechanism whose use can be determined by the community of practice.

A schema.org Dataset has several representations including JSON-LD. Using JSON-LD or another representation, Google Dataset Search "lets you find datasets wherever they're hosted, whether it's a publisher's site, a digital library, or an author's personal web page."



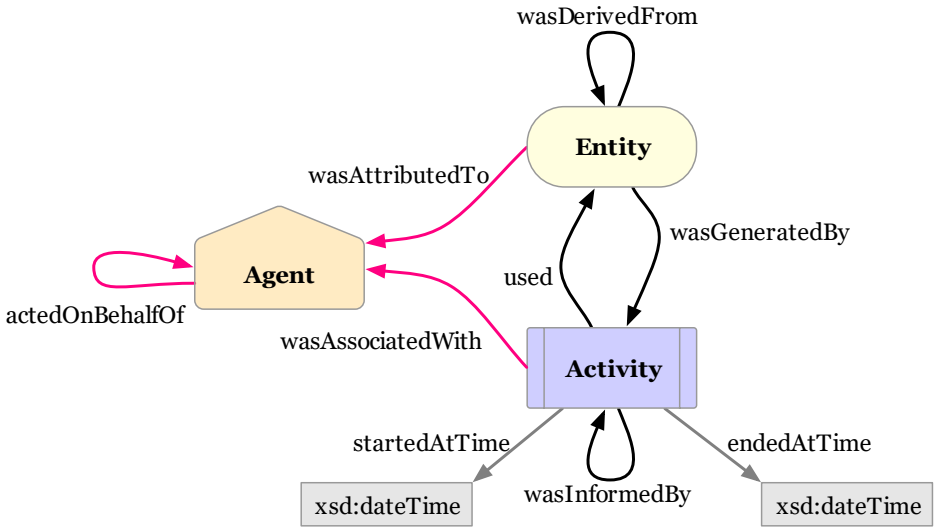
Note that a schema.org Dataset example markup represented in JSON-LD is included in the DDI - CDI Examples Document. This example is based on the Health and Demographic Surveillance Systems (HDSS) event dataset schema that is widely used across much of Sub-Saharan Africa for purposes of demographic surveillance.

3. PROV-O

a. *Relationship with DDI - CDI:* Trace relationship with Data Management

b. *Description*

In this specification, the standard states: “It provides a set of classes, properties, and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts. It can also be specialized to create new classes and properties to model provenance information for different applications and domains.”



The intention of the recommendation is to provide an extensible model, which can be applied to different domains and systems. DDI - CDI uses PROV-O in this way, by specializing specific classes to reflect those found in the DDI - CDI model.

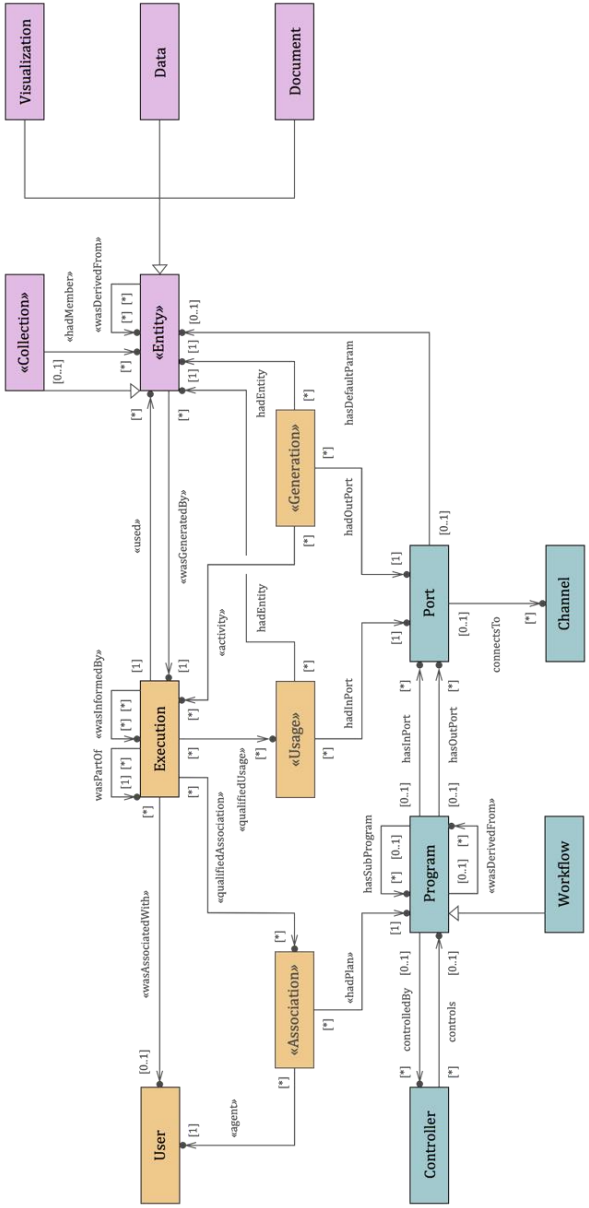
PROV-O is expressed as an OWL2 ontology, rather than as a UML model. Classes depicted in the DDI - CDI diagrams represent the corresponding classes as described in the PROV-O OWL2 definition, but the UML classes are created by the DDI - CDI to represent those classes – they are not a formalism which can be taken directly from the specification in UML form. In DDI - CDI, the trace relationships typically represent specializations of the more generic PROV-O classes, but would indicate that the properties and relationships of these classes can also be applied to the appropriate DDI - CDI objects.

4. ProvONE

a. *Relationship with DDI - CDI:* Trace relationship with Data Management

b. *Description*

Recently, PROV-O has been extended by ProvONE. ProvONE makes PROV-O data and computer program specific. In PROV-O entities didn't distinguish data at different level of specificity. And the PROV-O Plan entity lacked the specificity to describe the structure of computer programs and the specific successions of activities (workflows) that programs create. Here is the ProvONE Conceptual Model:


Visualization
Data
Document
«Collection»
«Entity»
«Generation»
«Usage»
«Association»
Execution
Port
Channel
Program
Workflow
User
Controller
«hadMember»
«wasDerivedFrom»
«used»
«wasGeneratedBy»
hadEntity
hasDefaultParam
hadOutPort
«activity»
hadInPort
connectsTo
«wasInformedBy»
wasPartOf
«qualifiedAssociation»
«qualifiedUsage»
hasInPort
hasOutPort
hasSubProgram
«wasAssociatedWith»
«hadPlan»
controlledBy
controls
«agent»

5. GSIM Concept Group (1.2)

a. *Relationship with DDI - CDI:* Trace relationship with Concepts and Variables (Foundational Metadata)

b. *Description*

GSIM provides a reference model of all the information – data, metadata, metrics, etc. – used in statistical production by national statistical agencies, and supra- and international statistical institutions. It is a product of the High-Level Group for the Modernization of Official Statistics (HLG-MOS), coordinated by the UN Economic Committee for Europe's Statistical Programme.

As a reference model, GSIM is primarily intended to facilitate more precise communication between implementers but does not serve directly as an implementation standard. It is possible that DDI - CDI can be used as a model for the implementation of some GSIM constructs. GSIM is modeled in UML, which makes the trace relationships function exactly as described in the UML specification quoted above. Correspondences between DDI - CDI classes and those in GSIM are often very direct, as they often model identical phenomenon with different levels of focus. Nuances of these relationships will be described on a class-by-class basis as appropriate in DDI - CDI.

GSIM consists of several groups including the Structure Group, the Business Group and the Concept Group. The GSIM Concept Group traces to Concepts and Variables (Foundational Metadata) in DDI - CDI. Both the DDI - CDI Variable Cascade and the Datum constructs in DDI - CDI extend the comparable classes from GSIM.

6. GSIM Structure Group (1.2)

a. *Relationship with DDI - CDI:* Trace relationship with Data Description

b. *Description*

See above for general information about the GSIM reference model. As described above, GSIM consists of several groups including the Structure Group, the Business Group and the Concept Group. The GSIM Structure Group traces to Data Description in DDI - CDI. Data Description adds new structure components to the group of structure components defined in the GSIM Structure Group enabling DDI - CDI to describe types of data that GSIM does not describe.

7. GSIM Business Group (1.2)

a. *Relationship with DDI - CDI:* Trace relationship with Data Management

b. *Description*

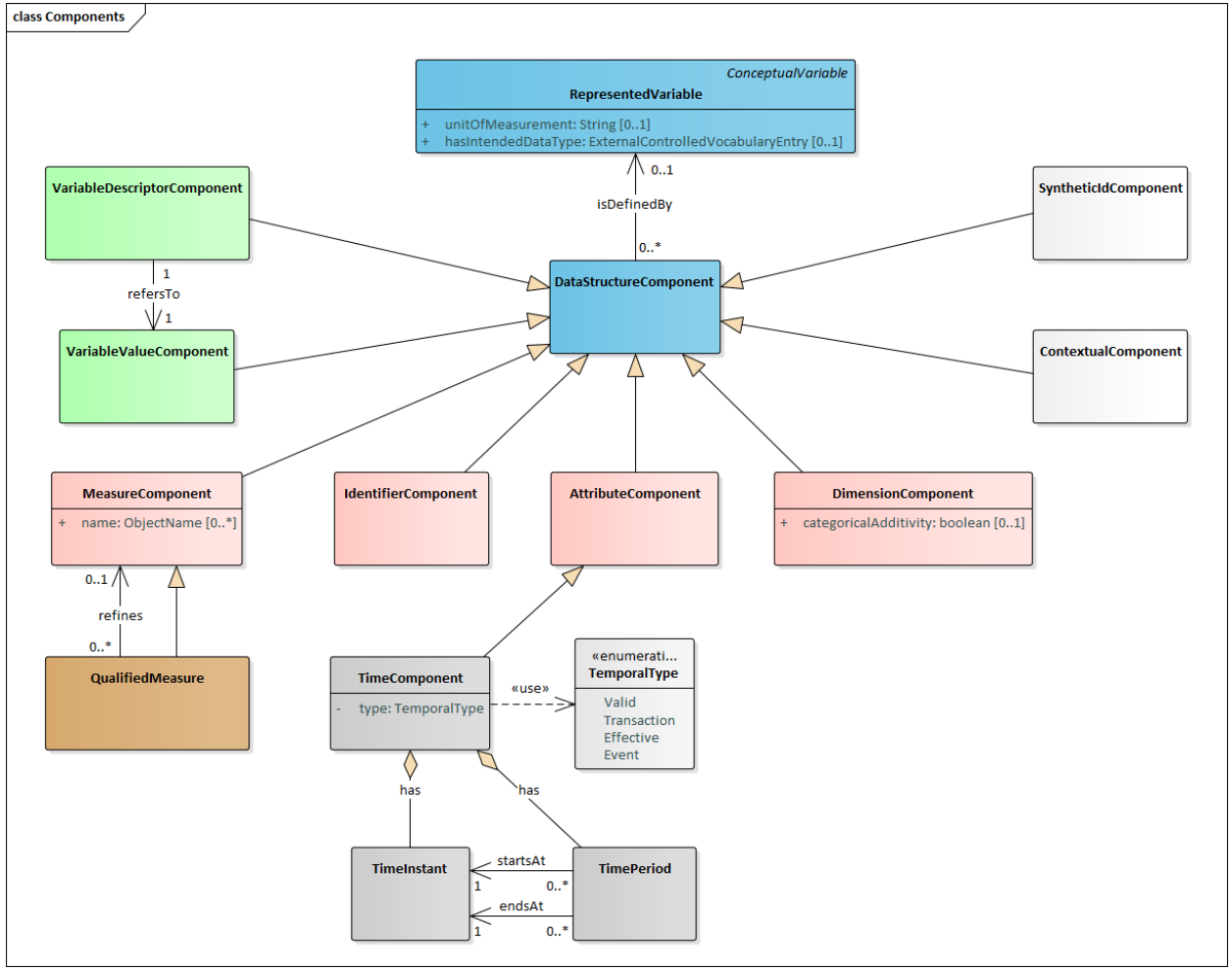
See above for general information about the GSIM reference model. As described above, GSIM consists of several groups including the Structure Group, the Business Group and the Concept Group. The GSIM Business Group traces mostly to Data Management in DDI - CDI. Data Management traces to PROV-O and ProvONE more than it does to the GSIM Business Group.

8. ISO 17369 Statistical Data and Metadata Exchange (SDMX)

a. *Relationship with DDI - CDI:* Map relationship with Data Description DimensionalDataSet

b. *Description:*

The current version of SDMX as of this writing is the 2.1 Consolidated version 2013. The SDMX Information Model provides a UML formalization against which the DDI - CDI model can be mapped. Recall from Document Two (the Detailed Model) that DDI - CDI has a rich set of data structure components that can be combined to define many dataset types including the dataset definitions supported by the SDMX Information Model.



A map between the DDI - CDI DimensionalDataSet and components of the SDMX Information Model is under development.

9. The RDF Data Cube Vocabulary (QB)

a. *Relationship with DDI - CDI:* Trace relationship with Data Description DimensionalDataSet

b. *Description:*

The RDF Data Cube Vocabulary is based on the SDMX 2.0 standard and covers the description of multi-dimensional data sets. It provides a set of classes and properties but does so without using any standard formalization. Consequently, the classes represented in the DDI - CDI model are created by DDI - CDI to reflect Data Cube classes, but do not come from any specific published UML model. They will carry with them the properties of the Data Cube classes when they appear in trace relationships with the DDI - CDI model.

10. RAIRD Information Model

a. *Relationship with DDI - CDI: :* Trace relationship with Data Description LongDataSet

b. *Description*

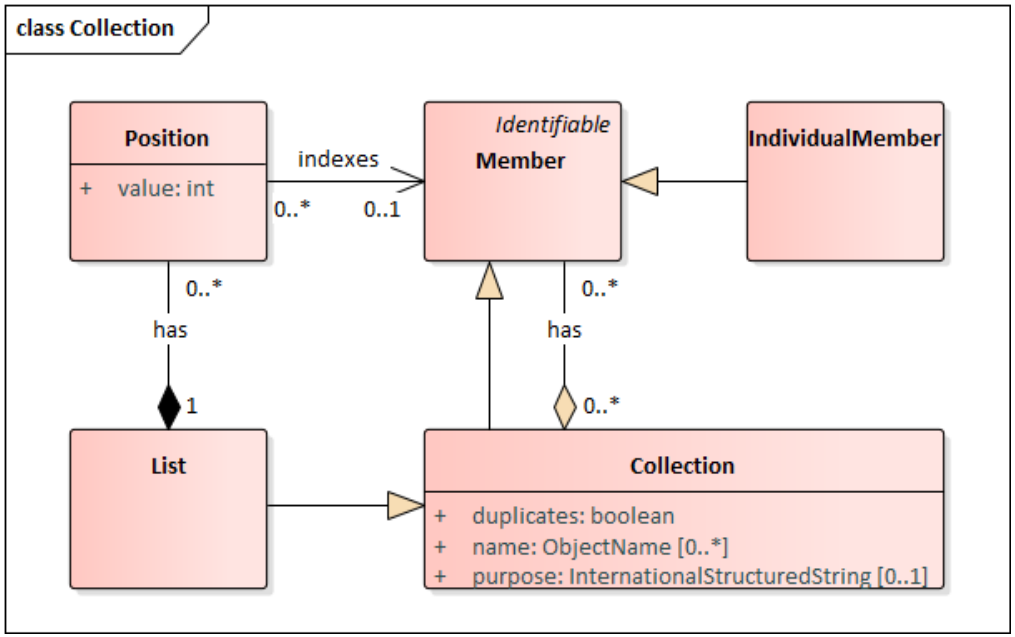
RAIRD supports the description of an event history dataset. In an event history dataset RAIRD builds on the GSIM datum-based data structure and adds an event period such that all observations, regardless of type, may be represented as follows:

Unit Identifier	Variable Reference	Value	Start Date	End Date
0937	DOB	1971-05-03	1971-05-03	-
0937	GENDER	1	1971-05-03	-
0937	MAR_STAT	M	2003-08-04	2010-02-02
0937	CIVIL_UNION_ID	4765	2003-08-04	2010-02-02
0937	MAR_STAT	D	2010-02-02	2012-02-02
0937	MAR_STAT	M	2012-02-14	-
0937	CIVIL_UNION_ID	5678	2012-02-14	-
2100	DOB	1972-03-05	1972-03-05	-
2100	GENDER	2	1972-03-05	-
2100	MAR_STAT	M	2012-02-14	-
2100	CIVIL_UNION_ID	5678	2012-02-14	-

III. Design Patterns

A. Using the Collections pattern

DDI-CDI introduces a generic Collections pattern that can be used to model different types of groupings and aggregations of objects, from simple unordered sets to all sorts of hierarchies, nesting and sequences.



A collection is basically a container, which could be either a set, i.e. unique elements or a bag, i.e. repeated elements. Collections can also be extended with richer semantic, e.g. generic, partitive, and instance, among others, to support a variety of DDI 3.x and GSIM structures, such as Node Sets, Schemes, Groups, sequences of Process Steps, etc. This pattern provides an abstraction to capture commonalities among a variety of seemingly disparate structures.

A Collection consists of zero, one or more Members. A Member could potentially belong to multiple Collections. Sets are defined by setting the *duplicates* property to *false,* bags by setting it to *true*. Membership in a Collection is maintained by a *has* aggregation.

List is an extension of Collection for sequentially ordered collections. It uses Position and its *value* property to indicate the location of a Member in the sequence. Note that Position does not extend from Identifiable because it's never managed independently from the List it belongs to.

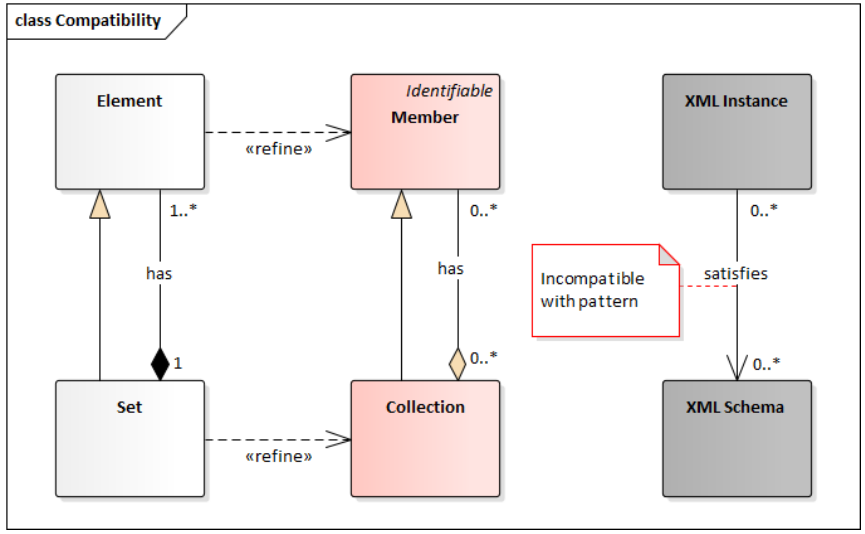
A Collection is also a Member, which allows for nesting of collections in complex structures. Members have to belong to some Collection, except in the case of nested collections where the top level is a member that doesn't belong to any collection. In addition, IndividualMember can be used to indicate that the member is not itself a Collection.

This pattern can be used via a special type of association called *refine*. DDI-Core uses *refine* to say that a group of class "behave" like the Collections pattern.

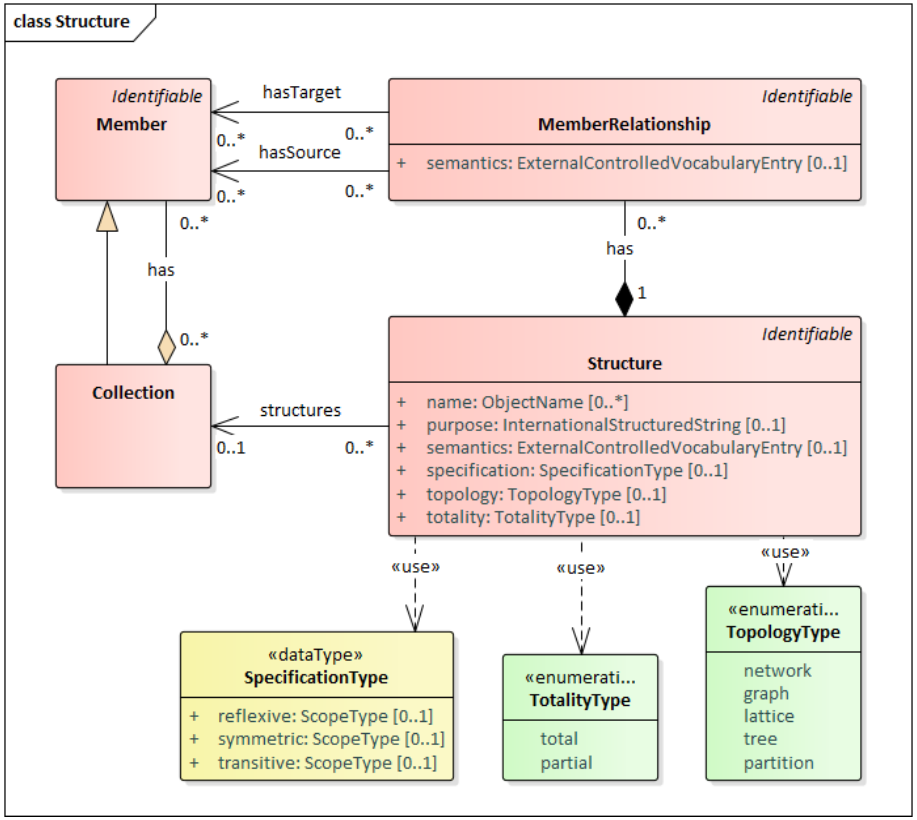
Refine is a sort of weak realization. To refine this pattern, all classes involved must be associated in a way that is compatible with the pattern. As a rule of thumb, a more restrictive type of association than the one that appears in the pattern is compatible, a looser one is not. For instance, since a Collection uses a *has* aggregation to identify its member, classes realizing the pattern need to be related by either an aggregation (same type) or a composition (more restrictive). In addition, the association has to be in the right direction, so that the class refining Collection is the "whole" (the diamond end) and the class refining

Member is the "part". Similar compatibility rules apply to cardinality. Furthermore, all associations must be refined, with the exception of *IsA* associations, which are usually part of the pattern definition and do not apply to individual refinement in the same way. Renaming properties and associations does not affect compatibility as long as the documentation clearly explains how they map to the association in the pattern.

For instance, consider the model diagram below. A Set is defined in this example as being composed of at least one Element, i.e. no empty Sets are allowed, and an Element always belong to one and only one Set. This is indicated by the cardinalities on the *has* association. Consequently,



deleting the Set will also delete its Elements. Such definition is compatible with the Collections pattern and thus Set and Element can refine Collection and Member, respectively. In contrast, Schema and XML Instance cannot refine the pattern because the Schema is not a grouping of XML Instances so the notion of a Collection being a container of Members doesn't hold.



Beyond the sequential ordering provided by List, the Collections pattern includes a Structure class that supports more complex structures and orderings of members via MemberRelationship.

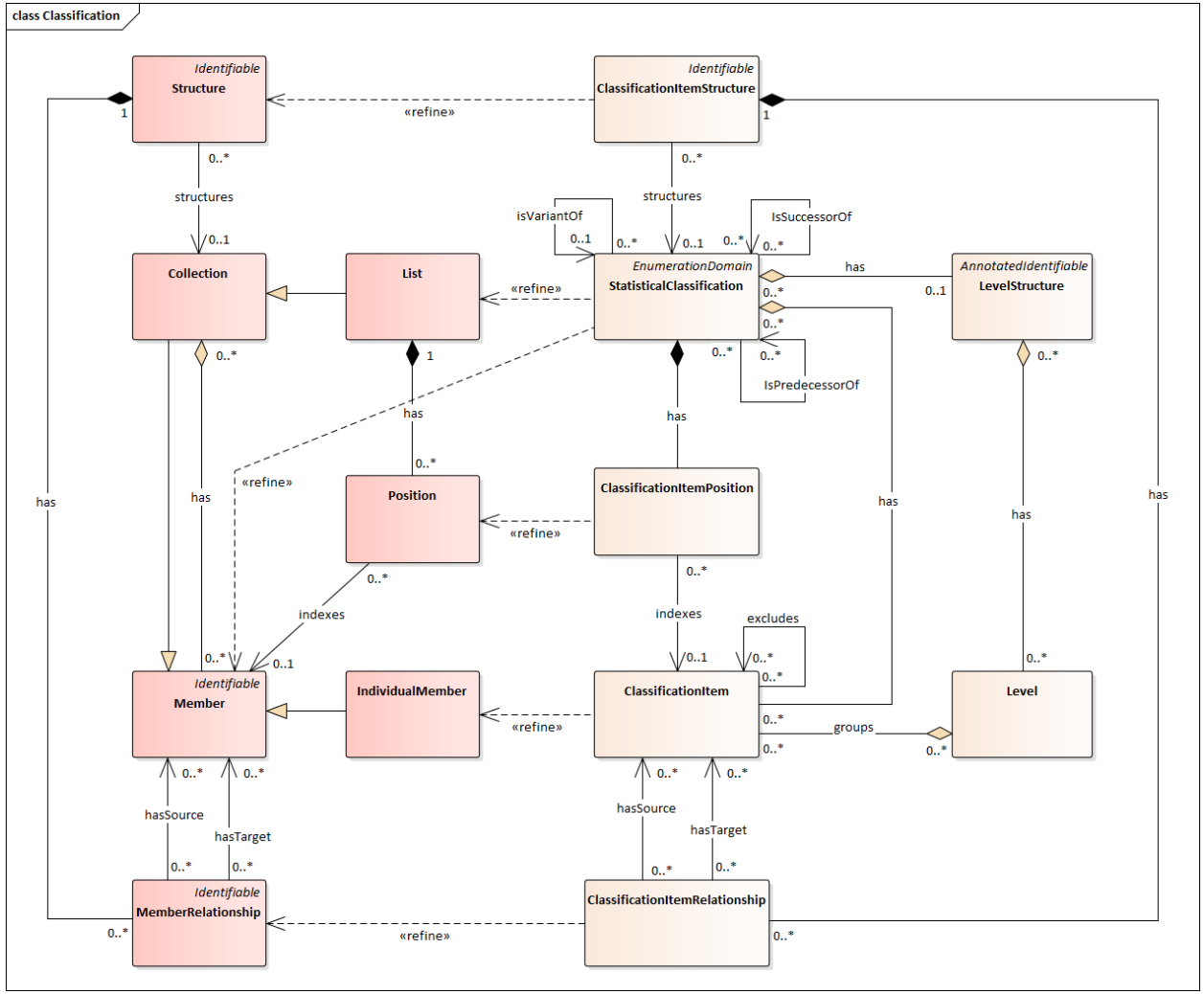
A Structure consists of one or more MemberRelationships, which are tuples linking Members at the end of the *hasSource* and *hasTarget* associations.

A Structure can have a *specification* property, e.g. *reflexive*, *symmetric*, and *transitive*, a *totality* property, e.g. *total* or *partial*, and a *topology* property, e.g. network, graph, lattice, tree, partition. These properties are defined by SpecificationType, TotalityType and TopologyType, respectively. A Structure can also have *semantics* defined by an ExternalControlledVocabularyEntry.

A Structure is said to be *total,* if all members of the associated collection are related to each other (otherwise, it is said to be *partial*); s*ymmetric*, if for any pair of members, *a*, *b* in the associated collection, whenever *a* is related to *b* then also *b* is related to *a* (otherwise, it is said to be *anti-symmetric*); r*eflexive,* if all members of the associated collection are related to themselves (otherwise, it is said to be *anti-reflexive*); and *transitive,* if for any members *a*, *b*, *c* in the associated collection, whenever *a* is related to *b* and *b* is related to *c* then *a* is also related to *c* (otherwise, it is said to be *anti-transitive*).

These characteristics can be combined to define different types of Structures, e.g. equivalence relations and partial order relations, among others. Equivalence relations are useful to define partitions and equivalence classes (e.g. Levels in a StatisticalClassification). Partial order relations can be used to represent lattices (e.g. class hierarchies, partitive relationships), parent-child relations can define trees and acyclic precedence relations can represent directed acyclic graphs (e.g. molecular interactions, geospatial relationships between regions).

Let us illustrate how this model works with a simple instance. Consider a North American Industry Classification System (NAICS) StatisticalClassification with ClassificationItems representing type of economic activity, such as *Mining*, *Manufacturing*, *Finance*, etc. The following diagram shows how the statistical classification classes refine the Collections pattern.



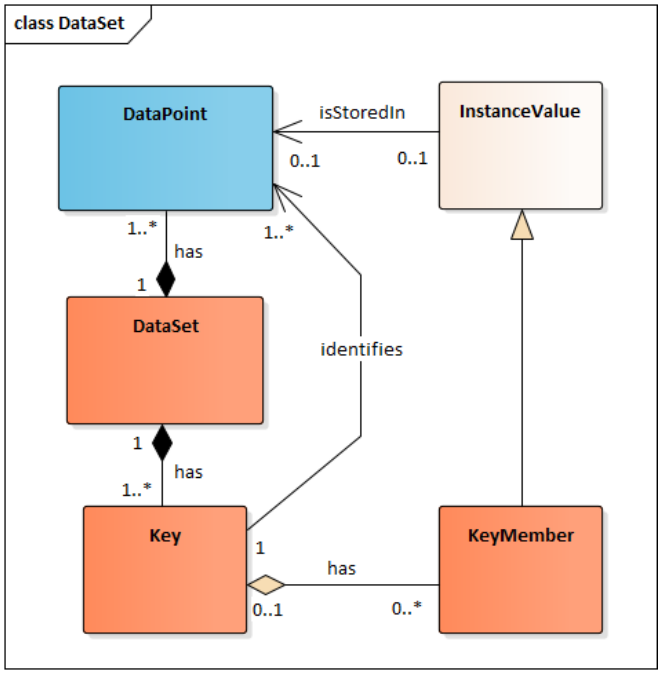
Note that StatisticalClassification refines Collection and ClassificationItem refines IndividualMember. This means we can view ClassificationItems such as *Manufacturing*, *Machinery manufacturing*, and *Educational services* in NAICS as Members organized in a hierarchy by a ClassificationItemStructure which consists of a set of ClassificationItemRelationships representing the parent-child relationships between

items. For instance, <*Manufacturing*, *Machinery manufacturing*> is a ClassificationItemRelationship in which *Manufacturing* is the source and *Machine manufacturing* is the target.

Note that by maintaining the hierarchy in a separate structure, i.e. ClassificationItemStructure, items can be reused in multiple classifications. For instance, a NAICS variant groups economic activities into two main industry groupings: the *goods-producing industries* and the *services-producing industries*. Because of the separation of hierarchy and categories, adding that high-level grouping doesn't require a change in the structure and definition of the underlying industry ClassificationItems.

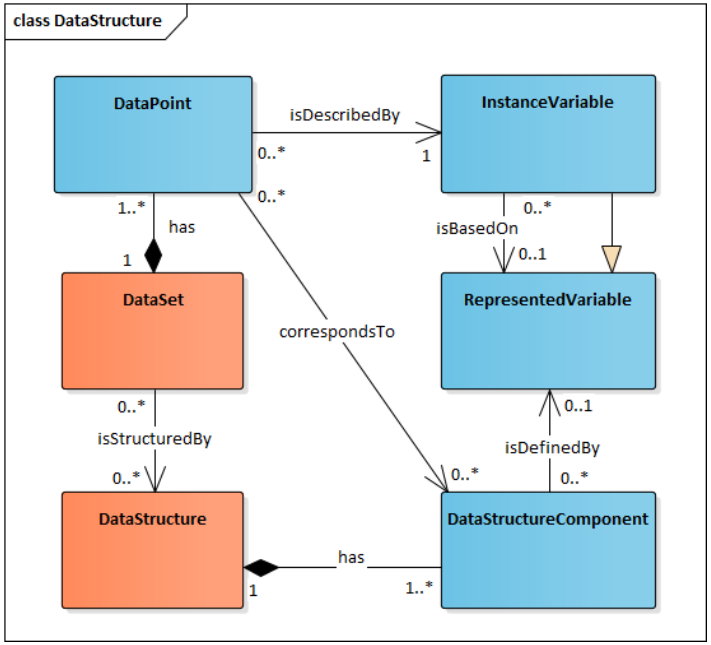
B. Using the Data Description pattern

Another pattern in DDI-CDI is data description. This pattern organizes data into datasets and their data structures. DDI-CDI includes four basic types of data sets: Key-Value, Wide, Long and Dimensional. All these types of data sets, and more, can be described with the same pattern.



A DataSet is an organized collection of data that consists of DataPoints and Keys. A DataPoint stores an InstanceValue, which is essentially a single data instance. Within DataSets, DataPoints are uniquely identified by Keys, which are collections of InstanceValues and as such they are also stored in other DataPoints. Each InstanceValue that forms part of a Key is called a KeyMember. For instance, a social insurance number and a date can be two key members that together form a key which identifies data points containing blood test results of a patient. The set of data points identified by a key constitutes a *record* or a *row* in rectangular data files and other traditional data structures. We don't have an explicit notion of a record in our model to support other types of flexible data organizations.

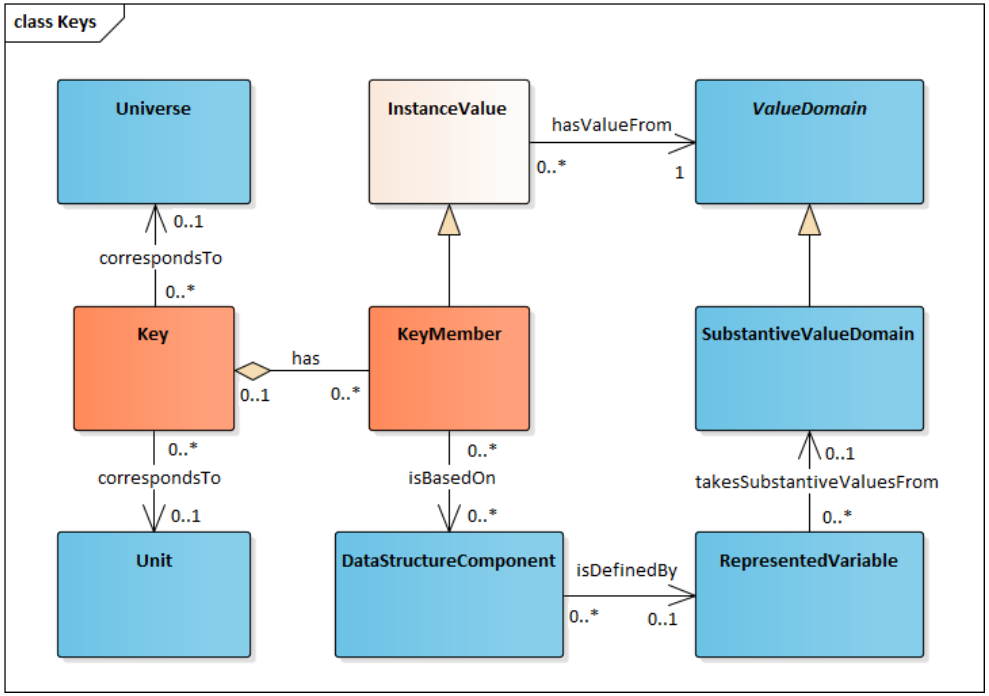
DataSets are further described by DataStructures. This model also supports schema on-read type of data description, in which data can be stored with basic or no descriptive information and then descriptions can be added as necessary at the time of use.



A DataStructure consists of multiple DataStructureComponents defined by RepresentedVariables. Essentially, a data structure component is the use of a represented variable in the context of a given data structure. These components could be of different types, the most common ones being *identifier*, *dimension*, *measure* and *attribute*. For instance, the same marital status variable can play the role of a *measure* in one data structure and a *dimension* in another. In such a case there are two DataStructure components, i.e. a MeasureComponent and a DimensionComponent, using the same marital status variable.

Each DataPoint in a DataSet corresponds to some DataStructureComponent in its associated DataStructure. Note that the same DataStructure can apply to multiple DataSets. Therefore, the InstanceVariable associated to each DataPoint needs to be related to the RepresentedVariable used by the DataStructureComponent that corresponds to that DataPoint.

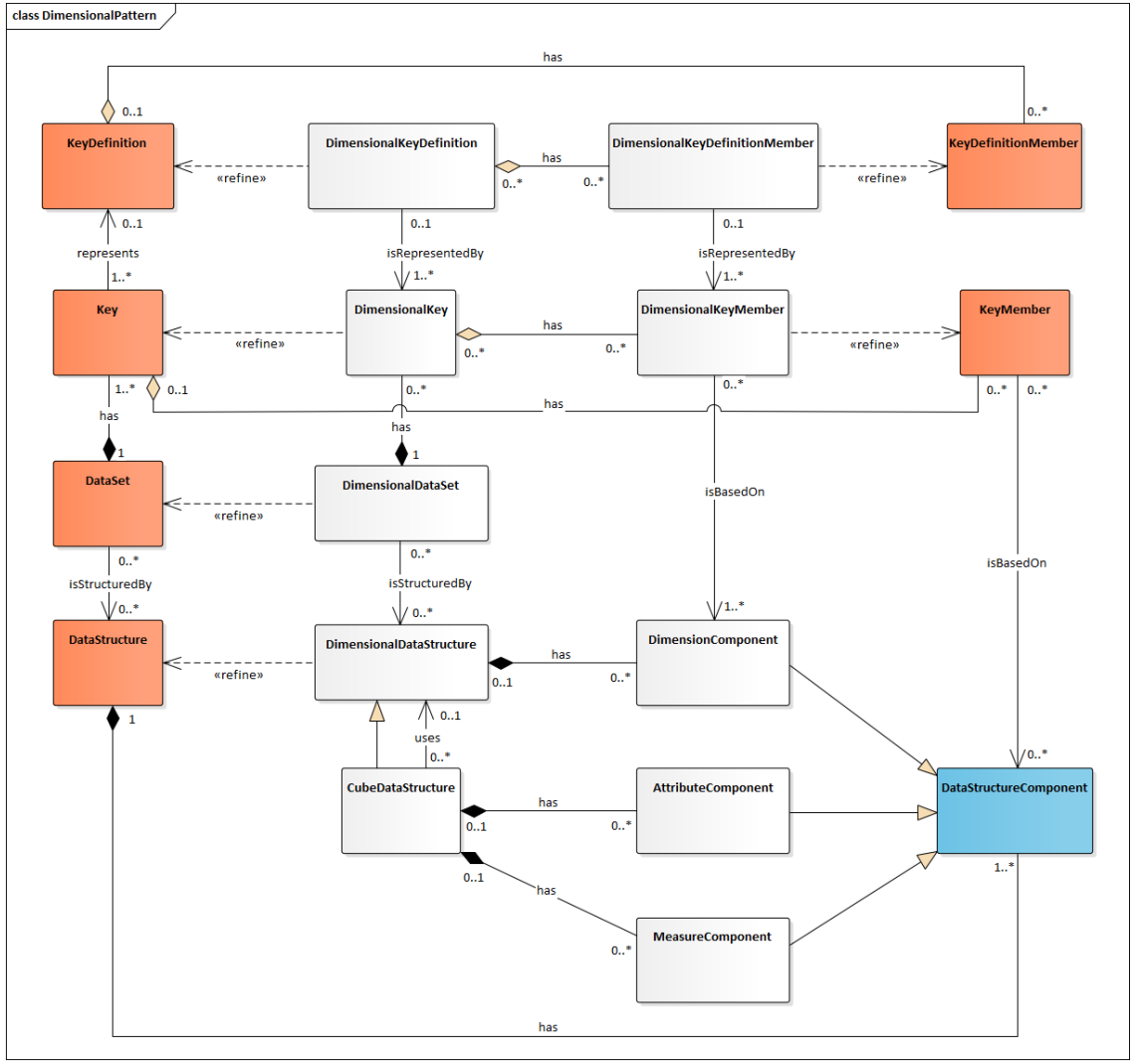
There is a similar relationship between KeyMembers and DataStructureComponents. Remember that KeyMembers are also InstanceValues and they are based on some DataStructureCompoment. As such, a KeyMember is a value from some ValueDomain, which is the same ValueDomain of the RepresentedVariable associated with the DataStructureComponent the KeyMember is based on. The diagram below shows these relationships.



In addition to identifying data points and relating to data structure components, a key is also related to either a Unit or a Universe. A Key corresponds to a Unit when the DataPoints they identify contain microdata whereas it corresponds to a Universe when they contain aggregate/macro data. Note that corresponding to a Unit (or a Universe) doesn't mean uniquely identifying it, since different Keys can correspond to the same Unit, like in a long DataSet where multiple rows correspond to measures related to the same Unit. The job of uniquely identifying Units (and Universes) rests on the IdentifierComponents (and the DimensionComponents, respectively). Keys uniquely identify only sets of DataPoints, which form records or rows.

As mentioned, Keys are groups of KeyMembers, which are InstanceValues coming from ValueDomains. As such, they usually have conceptual counterparts called KeyDefinition and KeyDefinitionMember, respectively. A KeyDefinition conceptually define the DataPoints a Key identify. They do that by grouping ConceptualValues (the KeyDefinitionMembers), which are concepts represented by the KeyMembers. In other words, Keys are representations of concepts in KeyDefinitions. These concepts come from the ConceptualDomain of the ConceptualVariable associated to the RepresentedVariable from whose ValueDomain KeyMembers take values from.

Let us see how this pattern works with an example. Consider a data cube that contains the average income and the total population by city. City is a DimensionComponent and average income and total population are MeasureComponents. They are grouped into a CubeDataStructure which structures a DimensionalDataSet. (CubeDataStructure is an extension of DimensionalDataStructure, which is a collection of dimensions that can be reused and maintained separately.) DimensionalDataStructure refines DataStructure and DimensionalDataSet refines DataSet.

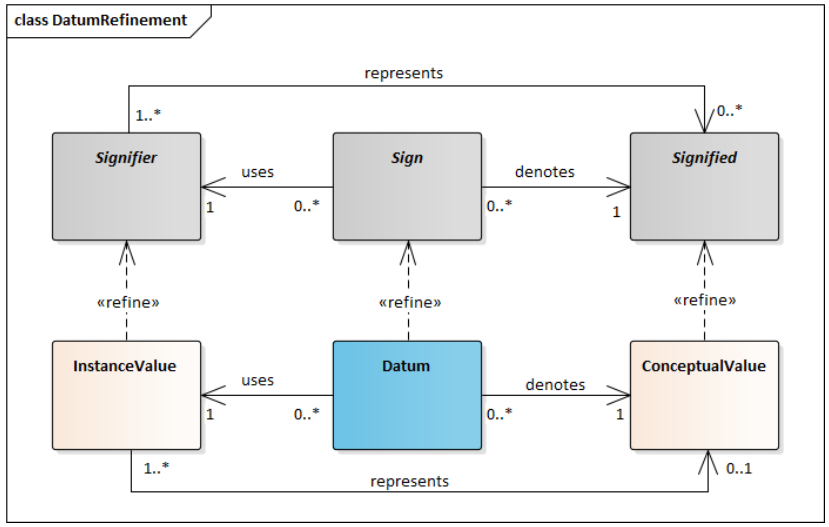


The rest of the pattern needs to be refined by the appropriate concrete classes. Key, KeyMember, KeyDefinition and KeyDefinitionMember are refined by DimensionalKey, DimensionalKeyMember, DimensionalKeyDefinition and DimensionalKeyDefinitionMember, respectively. Note that rather than being based on DataStructureComponent, a DimensionalKeyMember is based on DimensionComponent instead. That's a valid refinement of the pattern since DimensionComponent is an extension of DataStructureComponent. In this way, the refinement is more precise than if it were based on DataStructureComponent.

C. Using the Signification pattern

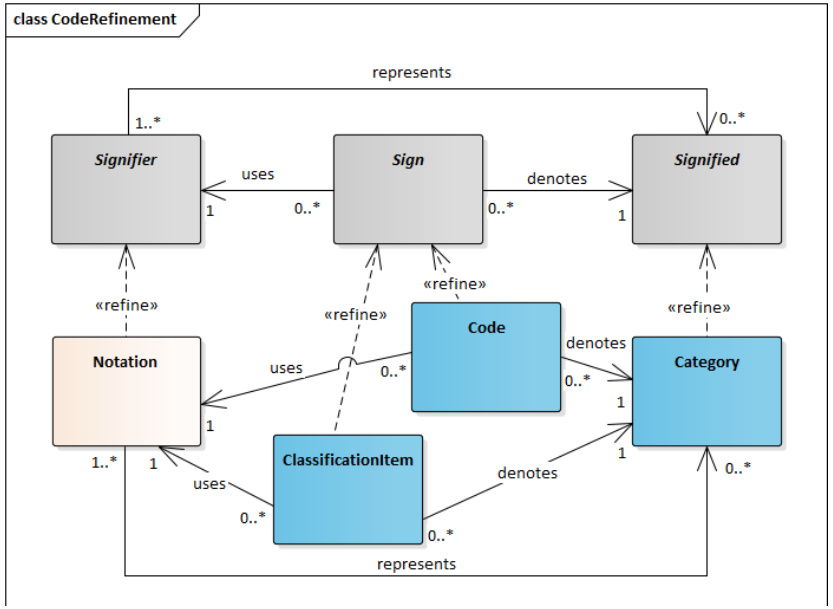
A Sign links a Signified with a Signifier that denotes it. A Signifier is a concept whose extension consists of tokens (perceivable objects). The Signified is the concept being represented by a Signifier. For instance, the concept of *integer five* is the Signified and all its different representations are tokens of its Signifier.

Signifier, Sign and Signified become part of the DDI-CDI signification pattern.



A Datum is an example of a Sign that links an InstanceValue with a ConceptualValue. An InstanceValue is a single data instance as it appears in a DataPoint. It's the representation of a concept, more precisely of a ConceptualValue, which is a concept with the notion of equality defined. This

notion of equality is important when representing data since data needs to be copied and compared, which is only possible with some equality operation. InstanceValue, ConceptualValue and Datum therefore refine Signifier, Signified and Sign, respectively.



The reason for making Signifier, Sign and Signified into a pattern to be refined as opposed to classes to be extended is that Concepts are not always Signifieds, which is what a specialization would imply. In fact, *a Concept is a Signified only if there is a Signifier that represents it*. The refinement means that the Concept is going to behave like a Signified only in the context of the pattern.

Another example of the use of the pattern is Code, which enters into the picture as a refinement of Sign. A Code then is a Sign that has Non-Linguistic Signifiers and where the Signified is a Category (Concept). ClassificationItem is a similar refinement of Sign linking a Notation to a Category. The Signifier in this case is Notation, which is just the representation of the Category within the context of a Code or a Classification Item.

IV. UML Subset

Please see the document "UML Class Diagram - Practioner's Subset for Data Modeling - Detailed List" in this package at `\DDI-CDI Public Review 1\1 Specification Documents\Supporting Documents`.

V. Design Notes and Modelling Approach

A. Introduction

DDI-CDI follows a model-driven approach. Model Driven Architecture® (MDA®) is an approach to software design, development and implementation spearheaded by the Object Management Group (OMG), a computer industry standards consortium. In the here used meaning, it starts with a model of data and the description of it (the application's business functionality and behavior) using Unified Modeling Language (UML). This model remains stable as technology evolves, extending and thereby maximizing software ROI (return on investment). Portability and interoperability are built into the architecture.

DDI-CDI uses a subset of the Class Diagrams of UML version 2. This subset of UML class diagram elements (see preceding section) is intended for data modeling. It focuses on core elements which are well known in object-oriented programming. The subset focuses on elements which describe classes, their interrelationships, and their attributes. This subset enables simple modeling, easy understanding, portability to many UML tools, and good mapping options to target representations. It is described in the document on "UML Class Diagram - Practioner's Subset for Data Modeling".

The specification UML Version 2.5.1[1] is used as canonical UML specification. The namespaces of UML 2.4.1 and XMI 2.4.1 are used in the representation as Canonical XMI. The version 2.4.1 is currently implemented in a larger number of UML tools. The namespaces of UML and XMI can be changed to the ones of 2.5.1. The used subset of UML is defined identically in the UML versions 2.4.1 and 2.5.1.

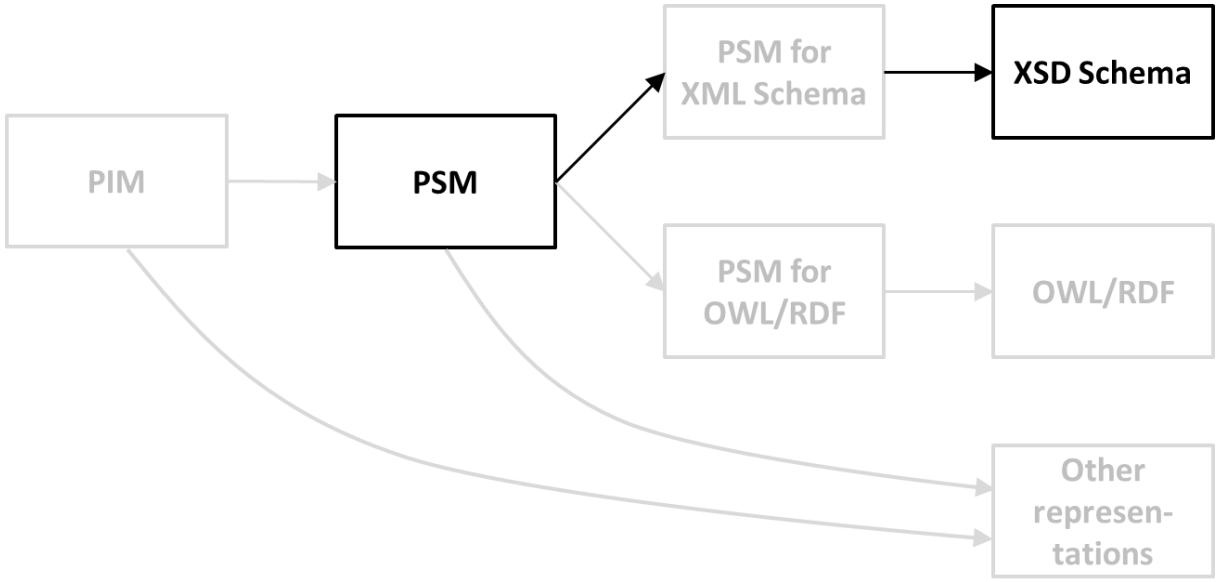
B. Type of Model

In model-driven architecture, a distinction is made between the platform independent model (PIM) and the platform-specific models (PSM). The PIM is translated to one or more PSMs.

[1] UML 2.5.1, https://www.omg.org/spec/UML/2.5.1/PDF

DDI-CDI uses this approach to make a distinction between the PIM (for conceptual purposes), the generic PSM (for multiple syntax representations with an object-oriented approach), and dedicated PSMs for specific syntax representations (encodings). All models are realized in UML. The PSMs are using the UML subset mentioned above.

PIM, PSM, and syntax representations in DDI-CDI (grey parts are not active yet):



Only the generic PSM exists currently and the derived XML Schema representation. The specific PSM for XML Schema is identical to the generic PSM.

The OWL/RDF representation is in the works.

The current model development was done in the generic PSM. It is planned for the future that the model development is done in the PIM which is then translated according to a set of business rules to the generic PSM. The differences between the future PIM and the generic PSM might comprehend for example navigability and multiplicity in associations.

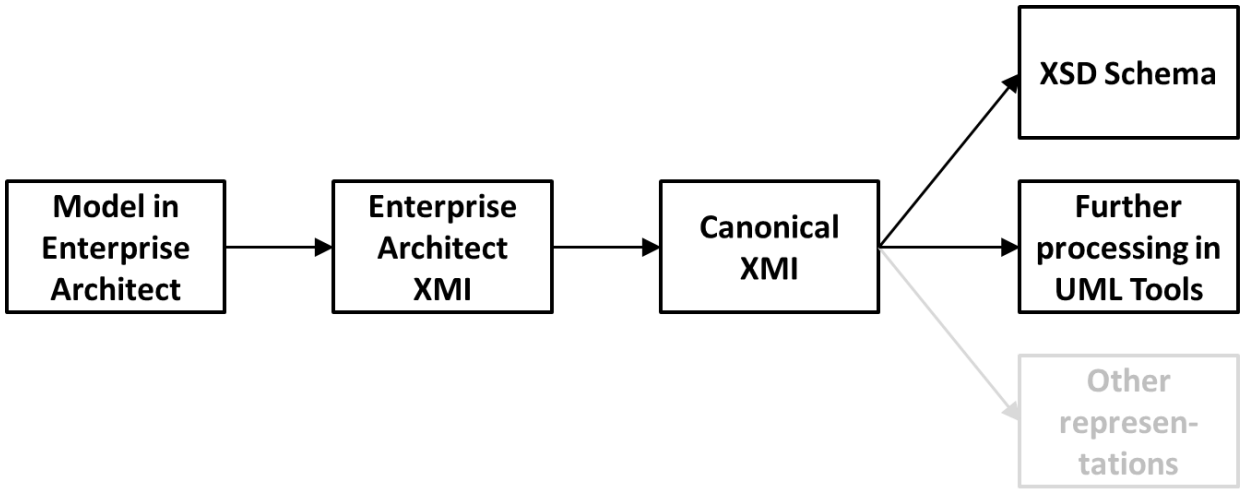
The UML tool Sparx Enterprise Architect[2] is used for the development of the generic PSM. The generic PSM is available as Enterprise Architect file. It can be viewed with the freely available viewer Enterprise Architect Lite[3].

C. Model Transformations

The transformation from the generic PSM to the XML Schema is done with XSLT programs. The Canonical XMI representation of the PSM is the input to these programs.

Enterprise Architect can export an own flavor of XMI. This is transformed by a XSLT to Canonical XMI.

Transformation chain:



D. Canonical XMI

DDI-CDI is available as Canonical XMI. This enables the import of the model into common UML tools. These tools can be used to analyze the model, to relate it to other UML models, and to generate syntax representations which are provided by these tools.

[2] Sparx Enterprise Architect, https://sparxsystems.com/products/ea/
[3] Enterprise Architect Lite, https://sparxsystems.com/bin/EALite.msi

The XML Metadata Interchange (XMI) is an Object Management Group (OMG) standard for exchanging metadata information via Extensible Markup Language (XML). The different vendors of UML tools have often XMI flavors which are specific to their tools. OMG proposed Canonical XMI to improve interoperability.

> "*Canonical XMI: A specific constrained format of XMI that minimizes variability and provides predictable identification and ordering. The constraints are detailed in Annex B.*"[4]

The Canonical XMI is available in two kinds one has the same association names as in the Enterprise Architect file, one unique association names across the model. The latter is intended for the transformation to XML Schema and for the import in some UML tools which complain about non-unique association names. The background is that strict UML requires unique names per item type in one package. The unique association names are constructed on the basis of the original association name plus the names of the connected classes.

E. Notes on Modeling

1. Structural Items

Package

- A package expresses a region of the interrelated content
- Packages (and classes) are named and organized in a way that they can be easily moved to another location of the model
    - The name of each package is a unique name (in the scope of all items) in the whole model

Class

A class can be understood as a blueprint for an object. It describes the type of objects for which DDI-CDI is a model for.

- Classes (and packages) are named and organized in a way that they can be easily moved to another location of the model
    - The name of each class must be a unique name (in the scope of all items) in the whole model
- Attributes are used
- Many classes have the attributes agency, id, and version. These items build a composite identifier aligned with the international registration data identifier (IRDI)[5]. Instantiated objects of these classes can be globally uniquely identified by these identifiers. This approach enables reuse of these objects on a granular level.

[4] XML Metadata Interchange (XMI) Specification, Version 2.5.1, https://www.omg.org/spec/XMI/2.5.1/PDF
[5] ISO/IEC 11179-6:2015, Information technology - Metadata registries (MDR) - Part 6: Registration, Annex A, Identifiers based on ISO/IEC 6523, http://standards.iso.org/ittf/PubliclyAvailableStandards/c060342_ISO_IEC_11179-6_2015.zip

Attribute

A class attribute is typed by a data type.

2. Relationships

Association

Only binary associations are used, i.e. two classes are related.

A notion of direction is used in DDI-CDI to be able to properly name associations so that they read as semantic triple (subject-predicate-object). Additionally, the direction is defined by a navigable association end at the "object" class. The association has unspecified navigability at the end of the "subject" class.

Associations are rendered in diagrams by connecting classes with a line. The navigable end is indicated by an open arrowhead (→) on one end of an association and owned by the class on the opposite end.

The definition of direction/navigation can be understood just as a recommendation, see also this citation from the official UML specification in the section on the semantics of associations.

> *„Navigability means that instances participating in links at runtime (instances of an Association) can be accessed efficiently from instances at the other ends of the Association. The precise mechanism by which such efficient access is achieved is implementation specific. If an end is not navigable, access from the other ends may or may not be possible, and if it is, it might not be efficient.*
>
> *NOTE. Tools operating on UML models are not prevented from navigating Associations from non-navigable ends."*[6]
>
> "*Specifying a direction of traversal does not necessarily mean that you can't ever get from objects at one end of an association to objects at the other end. Rather, navigation is a statement of efficiency of traversal.*"[7]

For specific uses, the direction of an association might not make sense. In this case, the navigation definition can be just ignored.

Association names should be unique in one package if possible. This is not always suitable in terms of achieving short names. Some UML tools comply in a strict sense to the UML rule that elements of related or the same type should have unique names within the enclosing package. For this purpose, a second representation of the model in Canonical XMI is provided which has unique association names per package.

[6] UML 2.5.1, Semantics of associations, page 200, https://www.omg.org/spec/UML/2.5.1/PDF

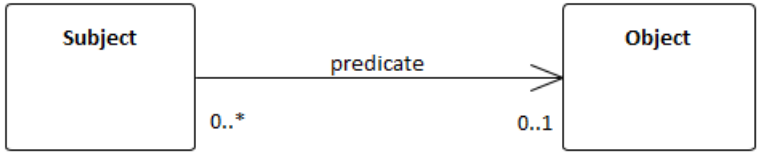
[7] The Unified Modeling Language User Guide, Booch, Grady; Rumbaugh, James; Jacobson, Ivar; Reading, Mass., 1999, page 144

Multiplicity

Multiplicity is formally defined as a lower and upper bound. Simply put: a multiplicity is made up of a lower and an upper cardinality. Cardinality is how many elements are in a set.

The default multiplicity of the "subject" class is 0..n. The multiplicity of the "object" class is usually 0..1 or 0..n. Zero for the lower cardinality allows flexibility in the process of producing metadata.

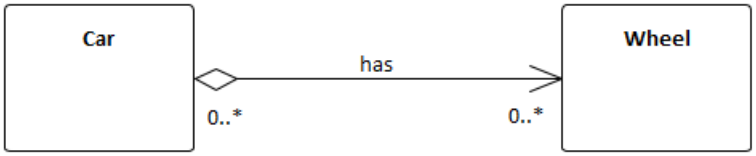
Association rendering:



Aggregation

Any semantics in aggregation are not seen which are not already covered by a common association with appropriate directed names, but it could provide a way of easily visualizing a whole/part relationship.

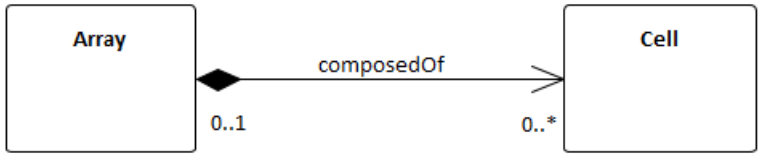
Aggregation rendering:



Composition

Composition is used for cases in which there is a strong lifecycle dependency, e.g. a cell in an array cannot exist without the array. However, it could provide a way of easily visualizing strong lifecycle dependency.

Composition rendering:

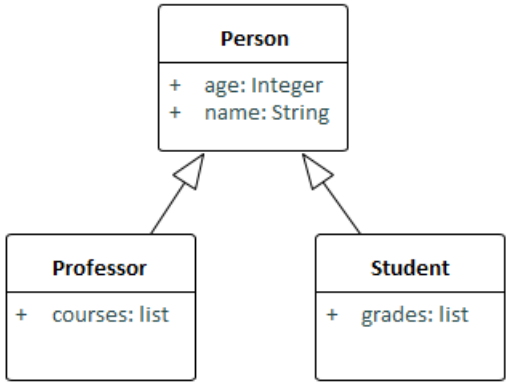


Generalization

A class can be an extension of another class (the general class). Attributes and associations of the general class are inherited. Only single inheritance is used, i.e. a class can only extend one other.

A data type can be an extension of another data type (the general data type). Attributes of the general data type are inherited. Only single inheritance is used, i.e. a data type can only extend one other. This applies also to primitive data types and enumerations.

Class generalization rendering:



3. Data Type Definition

Data Type

A data type can be a UML primitive data type, a structured data type, or an enumeration.

The UML primitive data types[8] are used: Boolean, Integer, Real, String, and UnlimitedNatural (the latter is only in XMI for the unlimited value of an upper cardinality).

[8] XMI representation of UML primitives, https://www.omg.org/spec/UML/20100901/PrimitiveTypes.xmi

A structured data type can have multiple attributes which are defined by other data types.

Some XML Schema primitive data types[9] are used. They are defined as UML primitive data types and defined semantically by the related XML Schema data type definition. Following XML Schema primitive data types are used: anyURI, date, and language.

All structured data types make finally use of the mentioned four UML primitive data types and three XML Schema primitive data types.

The UML primitive data types can be mapped to XML Schema data types in representations where they exist like in XML Schema and OWL/RDF.

Mapping of primitive data types[10]:

UML	XML Schema
PrimitiveTypes::Boolean	http://www.w3.org/2001/XMLSchema#boolean
PrimitiveTypes::Integer	http://www.w3.org/2001/XMLSchema#integer
PrimitiveTypes::Real	http://www.w3.org/2001/XMLSchema#double
PrimitiveTypes::String	http://www.w3.org/2001/XMLSchema#string
PrimitiveTypes::UnlimitedNatural	http://www.w3.org/2001/XMLSchema#string

Comment

Each item can have a definition which is expressed as UML comment.

4. Naming Convention

All items are named according to rules aligned with ISO/IEC 11179-5[11]. The names are possible compounds of multiple nouns and adjectives. Instead of a separator, the first letter of each name part within a single name is capitalized (sometimes called CamelCase).

Names of classes, data types, and enumeration literals start with an uppercase letter. Names of associations and attributes start with a lowercase letter.

[9] XML Schema primitive data types: https://www.w3.org/TR/xmlschema-2/#built-in-primitive-datatypes
[10] UML 2.5.1, XMI Serialization of the PrimitiveTypes model library, page 754, https://www.omg.org/spec/UML/2.5.1/PDF
[11] ISO/IEC 11179-5, Information technology - Metadata registries (MDR) — Part 5: Naming principles, http://standards.iso.org/ittf/PubliclyAvailableStandards/c060341_ISO_IEC_11179-5_2015.zip

F. Model Outline

The model outline of the Canonical XMI representation is listed below. These packages are intended for the users of the model. The main package “Classes” contains sub-packages with all classes for the three main areas foundational (Conceptual), data description (DataDescription), and process/provenance (Process) and supporting areas. The main package “DataTypes” contains sub-packages with all supporting data types.

- **DDICDILibrary**
  - Classes
    - Agents
    - Conceptual
    - DataDescription
      - Components
      - Dimensional
      - KeyValue
      - Long
      - Wide
    - FormatDescription
    - Identification
    - Miscellaneous
    - Process
    - Representations
  - DataTypes
    - Enumerations
    - StructuredDataTypes
    - XMLSchemaDataTypes

The model as Sparx Enterprise Architect file has two additional packages: “Diagrams” which holds all diagrams used in the specification document, and “DesignPatterns”, which supports mainly the specification developers.

- DDICDI
  - DDICDIModels
    - **DDICDILibrary (see above)**

- DesignPatterns
  - CollectionsPattern
  - DataDescriptionPattern
  - SignificationPattern
- Diagrams

VI. Appendixes

A. The DDI - CDI Upper Model Properties and their Sources by Property Group

Upper Model Property[12]	Property Group	Source(s)
about	About	schema.org CreativeWork,
abstract	About	schema.org CreativeWork, DC
accessMode	About	schema.org CreativeWork
accessModeSufficient	About	schema.org CreativeWork
accessRights	About	DC, DDI Codebook
accessibilityAPI	About	schema.org CreativeWork
accessibilityControl	About	schema.org CreativeWork
accessibilityFeature	About	schema.org CreativeWork
accessibilityHazard	About	schema.org CreativeWork
accessibilitySummary	About	schema.org CreativeWork

[12] Generally speaking, the names of the properties have been adopted from schema.org's CreativeWork. In the case that there is a single source other than CreativeWork, the name of the property comes from the single source. One exception are properties whose names begin with the prefix *sd*. *sd* is an abbreviation for structured data in CreativeWork. Structured data refers not to what a CreativeWork is about – the Dataset context -- but the CreativeWork object itself. Codebook also makes this distinction. Some Codebook only properties have been added to the *sd* series. Definitions for each of the properties are linked. The link goes to DCMI Metadata Terms when it is the origin or it has greater specificity. Otherwise the link goes to CreativeWork in schema.org. When a property is sourced both to CreativeWork and DC (Dublin Core), review the Dublin Core / DDI Core Upper Model Map in this Appendix (C) to determine the nuances of these relationships.

Upper Model Property[12]	Property Group	Source(s)
accountablePerson	Credits	schema.org CreativeWork, DC
accrualMethod	About	DC
accrualPeriodicity	About	DC. DDI Codebook
accrualPolicy	About	DC
aggregateRating	Buzz	schema.org CreativeWork
alternativeHeadline	About	schema.org CreativeWork, DC
associatedMedia	About	schema.org CreativeWork, DC
audience	About	schema.org CreativeWork, DC
audio	About	schema.org CreativeWork, DC
author	Credits	schema.org CreativeWork, DC
award	Buzz	schema.org CreativeWork
character	About	schema.org CreativeWork
citation	About	schema.org CreativeWork, DC
citationRequirement	About	DDI Codebook
comment	About	schema.org CreativeWork
commentCount	Buzz	schema.org CreativeWork
concept	About	DDI Codebook
conditionsOfAccess	About	schema.org CreativeWork, DC
confidentiality	About	DDI Codebook
contentLocation	About	schema.org CreativeWork, DC
contentRating	Buzz	schema.org CreativeWork
contentReferenceTime	About	schema.org CreativeWork, DC
contributor	Credits	schema.org CreativeWork, DC
copyrightHolder	Credits	schema.org CreativeWork, DC
copyrightYear	Credits	schema.org CreativeWork, DC
correction	About	schema.org CreativeWork, DC
creativeWorkStatus	About	schema.org CreativeWork
creator	Credits	schema.org CreativeWork, DC
dataQualityMetrics	About	DDI Codebook
date	About	schema.org CreativeWork, DC
dateAccepted	About	schema.org CreativeWork, DC

Upper Model Property[12]	Property Group	Source(s)
dateCopyrighted	About	schema.org CreativeWork, DC
dateCreated	Credits	schema.org CreativeWork, DC
dateModified	About	schema.org CreativeWork, DC
datePublished	About	schema.org CreativeWork, DC
dateSubmitted	About	schema.org CreativeWork, DC
disclaimer	About	DDI Codebook
discussionURL	Buzz	schema.org CreativeWork, DC
editor	Credits	schema.org CreativeWork, DC
educationalAlignment	Buzz	schema.org CreativeWork, DC
educationalUse	Buzz	schema.org CreativeWork
encoding	About	schema.org CreativeWork, DC
encodingFormat	About	schema.org CreativeWork, DC
estimateOfSamplingError	About	DDI Codebook
exampleOfWork	About	schema.org CreativeWork, DC
expires	About	schema.org CreativeWork, DC
funder	Credits	schema.org CreativeWork, DC
genre	About	schema.org CreativeWork, DC
hasPart	About	schema.org CreativeWork, DC
headline	About	schema.org CreativeWork, DC
identifier	About	schema.org CreativeWork, DC
inLanguage	About	schema.org CreativeWork, DC
interactionStatistic	Buzz	schema.org CreativeWork, DC
interactivityType	Buzz	schema.org CreativeWork, DC
isAccessibleForFree	Buzz	schema.org CreativeWork, DC
isBasedOn	About	schema.org CreativeWork, DC
isFamilyFriendly	Buzz	schema.org CreativeWork
isPartOf	About	schema.org CreativeWork, DC
keyWords	About	schema.org CreativeWork, DC
learningResourceType	About	schema.org CreativeWork
license	Buzz	schema.org CreativeWork, DC
locationCreated	About	schema.org CreativeWork, DC

Upper Model Property[12]	Property Group	Source(s)
mainEntity	About	schema.org CreativeWork, DC
material	About	schema.org CreativeWork, DC
materialExtent	About	schema.org CreativeWork, DC
mentions	About	schema.org CreativeWork, DC
notesOnDataCollection	About	DDI Codebook
offers	Buzz	schema.org CreativeWork
position	About	schema.org CreativeWork
producer	Credits	schema.org CreativeWork, DC
provider	Credits	schema.org CreativeWork, DC
publication	Buzz	schema.org CreativeWork, DC
publisher	Credits	schema.org CreativeWork, DC
publisherImprint	Credits	schema.org CreativeWork
publishingPrinciples	Credits	schema.org CreativeWork
recodingAndDerivation	About	DDI Codebook
recordedAt	About	schema.org CreativeWork, DC
releasedEvent	Buzz	schema.org CreativeWork, DC
researchQuestion	About	DDI - CDI ResearchProgram and ResearchComponent
researchHypothesis	About	DDI - CDI ResearchProgram and ResearchComponent
responseRate	About	DDI Codebook
review	Buzz	schema.org CreativeWork
schemaVersion	About	schema.org CreativeWork
sdDatePublished	About	schema.org CreativeWork
sdIdentifier	About	DDI Codebook
sdLicense	About	schema.org CreativeWork
sdPublisher	About	schema.org CreativeWork
sdVersion	About	DDI Codebook
sdVersionNotes	About	DDI Codebook
security	About	DDI Codebook
sourceOrganization	Credits	schema.org CreativeWork

Upper Model Property[12]	Property Group	Source(s)
spatial	About	schema.org CreativeWork, DC
spatialCoverage	About	schema.org CreativeWork
sponsor	Credits	schema.org CreativeWork, DC
studyType	About	DDI - CDI ResearchProgram and ResearchComponent
temporal	About	schema.org CreativeWork, DC
temporalCoverage	About	schema.org CreativeWork
text	About	schema.org CreativeWork
thumbnailURL	About	schema.org CreativeWork, DC
timeRequired	About	schema.org CreativeWork
translationOfWork	About	schema.org CreativeWork
translator	Credits	schema.org CreativeWork, DC
typicalAgeRange	Buzz	schema.org CreativeWork
unitOfAnalysis	About	DDI Codebook
universe	About	DDI Codebook
version	About	schema.org CreativeWork, DC
video	About	schema.org CreativeWork, DC
weighting	About	DDI Codebook
workExample	About	schema.org CreativeWork, DC
workTranslation	About	schema.org CreativeWork, DC

B. DDI Codebook / DDI - CDI Upper Model Map

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
1. Metadata Production				
		Producer	sdPublisher	Codebook is broader
		Production Date	sdDatePublished	Codebook is broader
		DDI Document Version	sdVersion	Exact match
		Version Notes	sdVersionNotes	Exact match
		DDI Document ID Number	sdIdentifier	Exact match
2. Study Description				
	Identification			
	2.1.1.1	Title	Headline	Exact match
		Alternative Title	alternativeHeadline	Exact match
	2.1.1.4	Translated Title	translationOfWork	Close match
	2.1.1.5	ID Number	identifier	Exact match
	2.1.5.1	Study Type	typeOfResearch (ResearchProgram, ResearchComponent)	Codebook is broader
	2.1.5.2	Series Information	isPartOf, disambiguatingDescription	Codebook is broader
	Version		version	Exact match
	2.1.6.1	Production Date	datePublished	Close match
	2.1.6.3	Notes		not found in schema.org

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
	Overview			
	2.3.2	Abstract	abstract	Exact match
	2.2.3.8	Kind of Data	genre	Codebook is narrower
	2.2.3.6	Unit of Analysis	unitOfAnalysis	Exact match
	Scope			
	2.2.4	Description of Scope	identifier extension (PropertyValue)	Codebook is narrower
	2.2.1.2	Topics Classifications	identifier extension (PropertyValue)	Codebook is narrower
	2.2.1.1	Keywords	keywords	Exact match
	Coverage			
	2.3.1.3	Country	spatialCoverage, contentLocation	Codebook is broader
	2.3.1.4	Geographical Coverage	spatialCoverage, contentLocation, locationCreated	Codebook is broader
	2.2.3.7	Universe	identifier extension (PropertyValue)	Codebook is narrower
	Producers and Sponsors			
	2.1.2.1	Investigators	author, creator, accountablePerson	Codebook is broader
	2.1.3.1	Other Producers	Contributor, editor	Codebook is broader
	2.1.3.6	Funding	Funder	Exact match
	2.1.2.2	Other Acknowledgements	Contributor, sponsor	Codebook is broader

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
	2.4.1.2	Study Site	contentLocation	Codebook is narrower
	Sampling			
	2.3.1.4	Sampling Procedure	potentialAction	Codebook is narrower
	2.3.1.5	Deviations from Sample Design	correction	Codebook is narrower
	2.3.3.1	Response Rates	responseRate	Exact match
	2.3.1.12	Weighting	weighting	Exact match
	Data Collection			
	2.2.3.2	Dates of Collection	accrualMethodology, accrualPeriodicity, accrualPolicy	Codebook is broader
	2.3.1.6	Mode of Data Collection	potentialAction	Codebook is narrower
	2.3.1.3	Frequency of Data Collection	accrualMethodology, accrualPeriodicity, accrualPolicy	Codebook is broader
	2.3.1.9	Notes on Data Collection	notesOnDataCollection	Exact match
		Questionnaires	measurementTechnique	Codebook is narrower
	2.3.1.2	Data Collectors	contributor	Codebook is narrower
	Data Processing			
	2.3.1.13	Data Editing	correction	Codebook is narrower
	2.3.2	Other Processing	potentialAction	Codebook is narrower

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
	Data Appraisal			
	2.3.3.2	Estimate of Sampling Error	estimateOfSamplingError	Exact match
	2.3.3.3	MEIRU Data Quality Metrics	dataQualityMetrics	Exact match
	Data Access			
	2.4.2.4	Access Authority	provider	Codebook is narrower
	2.4.2.1	Confidentiality	confidentiality	Exact match
	2.4.2.7	Access Conditions	conditionsOfAccess, license	Codebook is broader
	2.4.2.5	Citation Requirement	citationRequirement	Exact match
	Disclaimer and Copyright			
	2.4.2.8	Disclaimer	disclaimer	Exact match
	2.1.3.2	Copyright	copyrightYear, copyrightHolder	Codebook is broader
	2.2.12	Contacts	provider, accountablePerson	Codebook is narrower
	2.1.4.2	Contact Persons	provider, accountablePerson	Codebook is narrower
3. File Description				
	Data Files			
	3.1.2	Contents	description	captured under study description
	3.1.7	Producer	author, creator	captured under study description

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
	3.1.10	Missing Data	notesOnDataCollection	Codebook is narrower
	3.2	Notes	notesOnDataCollection	Codebook is narrower
4. Variables Description				
			variableMeasured	Exact match
	4.2.15	Definition	description	Codebook is narrower
	4.2.12	Universe	identifier extension (PropertyValue)	Codebook is narrower
	4.2.21	Concepts	identifier extension (PropertyValue)	Codebook is narrower
	Question			
	4.2.8.1	Pre-Question Text	measurementTechnique	Codebook is narrower
	4.2.8.2	Literal Question	measurementTechnique	Codebook is narrower
	4.2.8.3	Post-Question Text	measurementTechnique	Codebook is narrower
	4.2.8.6	Interviewer Instructions	measurementTechnique	Codebook is narrower
	Derivation			
	4.2.19	Recoding and Derivation	recodingAndDerivation	Exact match
	Security			
		Security	conditionsOfAccess, confidentiality	Codebook is broader

Section	Subsection	Nesstar Publisher	Upper Model schema.org Dataset	Comparison
5. External Resources				
	Resource Description		citation	citation is its own ResearchComponent in a ResearchProgram with the same properties as a Codebook
		Type	about	Codebook is narrower
		Title	headline	Exact match
		Subtitle	alternativeHeadline	no equivalent in Schema.org
		Author(s)	author, creator	Codebook is broader
		Date	dateCreated	Codebook is broader
		Country	contentLocation	Codebook is broader
		Language	inLanguage	Exact match
		Format	encodingFormat	Codebook is broader
		ID Number	identifier	Exact match
	Contributor(s) and Rights			
		Contributor(s)	contributor	Exact match
		Publisher(s)	publisher	Exact match
		Rights	accessRights	Exact match

C. Another Codebook / Core Map

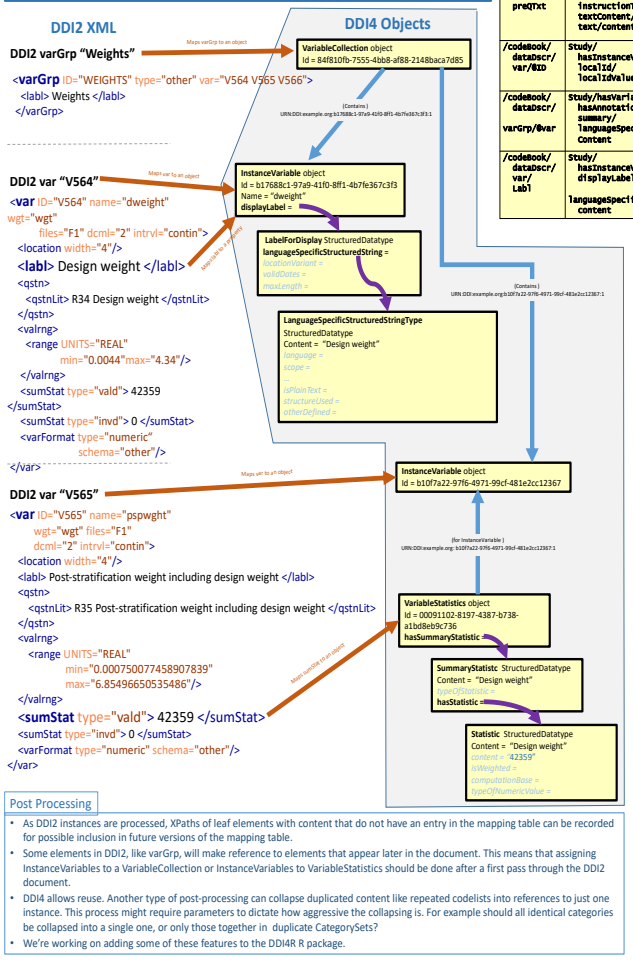
Please note that the name for DDI – CDI while under development was “DDI 4”.

The poster which appears here is difficult to read in this format, but is available online at Mapping DDI2 to DDI4.

Mapping DDI2 to DDI4, Larry Hoyle and Joachim Wackerow

- This poster describes a mapping of **leaf** DDI2 elements into corresponding DDI4 **properties**. The mapping can be described as a multi-column, **machine actionable, table**.
- DDI2 and the later versions differ in the underlying reliance on reusable objects. This makes mapping from DDI2 to DDI4 complicated. Software using this table can also collapse content that is repeated in DDI2 (like CodeLists) into single **reused DDI4 objects**.
- The table is derived from a spreadsheet listing DDI2 elements important to CESSDA. That table was further refined at the DDI4 Norway Sprint. The table below right explains the role of each column in the mapping table. Those columns contain the information needed to move information from one model to the other. Examples in the table below are color coded to corresponding cells in the table above.

The example below is taken from the European Social Survey file ESS1e06.6.xml



Post Processing

- As DDI2 instances are processed, XPaths of leaf elements with content that do not have an entry in the mapping table can be recorded for possible inclusion in future versions of the mapping table.
- Some elements in DDI2, like varGrp, will make reference to elements that appear later in the document. This means that assigning InstanceVariables to a VariableCollection or InstanceVariables to VariableStatistics should be done after a first pass through the DDI2 document.
- DDI4 allows reuse. Another type of post-processing can collapse duplicated content like repeated codelists into references to just one instance. This process might require parameters to dictate how aggressive the collapsing is. For example should all identical categories be collapsed into a single one, or only those together in duplicate CategorySets?
- We're working on adding some of these features to the DDI4R R package.

Sample rows from the mapping table. The full table currently lists 181 DDI25 XPaths.
This table can grow as new element types are found in DDI2 instances.

DDI25	DDI4PropertyName	IdentifiablesMapping	AbstractSubstitution	ParameterValues	IsIdRef	DDI4IdRefPath	DDI4IdRefAppend	Notes
/codeBook/ dataDscr/ var/ anlysunit	Study/ hasInstanceVariable/ usesUnitType/ Definition	/codeBook:Study, /codeBook/ dataDscr/ var:InstanceVariable, /codeBook/dataDscr/ var/anlysunit:UnitType						
/codeBook/ dataDscr/ var/ qstn/ preQTxt	Study/ hasInstanceVariable/ sourceCapture/ hasInstruction/ instructionText/ textContent/ text/content	/codeBook:Study, /codeBook/dataDscr/ var:InstanceVariable, /codeBook/dataDscr/var/ qstn:RepresentedQuestion, /codeBook/dataDscr/var/ qstn:Instruction	Capture: RepresentedQuestion, DynamicTextContent: LiteralText	RepresentedQuestion/ hasInstruction/ InstructionText/ textContent/ purpose/ languageSpecificStructuredString/ content='PreQuestionText'				
/codeBook/ dataDscr/ var/@ID	Study/ hasInstanceVariable/ localId/ localIdValue	/codeBook:Study, /codeBook/ dataDscr/ var:InstanceVariable			FALSE			Program will need to match DDI2 variable ID to DDI4 DdiUrn for references to the variable like varGrp
/codeBook/ dataDscr/ varGrp/@var	Study/hasVariableCollection/ hasAnnotation/ summary/ languageSpecificString/ Content	/codeBook:Study, /codeBook/ dataDscr/ varGrp:VariableCollection		Study/ hasVariableCollection/ hasAnnotation/ summary/ languageSpecificString/ scope='DDI2 variable IDs'	TRUE	contains/ member	TRUE, FALSE	requires matching DDI2 ID with DDI4 DdiUrn and inserting DDI4URN references into the VariableCollection
/codeBook/ dataDscr/ var/ Labl	Study/ hasInstanceVariable/ displayLabel/ languageSpecificStructuredString/ content	/codeBook:Study, /codeBook/dataDscr/ var:InstanceVariable						

Column Descriptions for the Table Above

Column	Function	Example	Details
DDI25	The unpredicated XPath of a DDI2 text or attribute node	/codeBook/dataDscr/var/anlysunit	Each piece of information to be imported from DDI2 should have a corresponding XPath listed.
DDI4PropertyName	The corresponding path to a leaf in DDI4	Study/hasInstanceVariable/ usesUnitType/definition	The first node in this path is a DDI4 class. The remaining nodes are properties in a chain down to a leaf value. The value of some properties are references to other objects, that object may need to be created. Other values are "structured datatypes".
IdentifiablesMapping	This maps a DDI2 sub-path to a DDI4 Identifiable class. An object of that class will need to be created for each unique instance of that DDI2 sub-path.	/codeBook:Study, /codeBook/dataDscr/var:InstanceVariable , /codeBook/dataDscr/var/anlysunit:UnitType	In the example to the left, for each unique var element in a DDI2 instance, the same DDI4 InstanceVariable needs to be used. A predicated XPath identifies a specific DDI4 object, e.g. /codeBook[1]/dataDscr[2]/var[7] indicates a sepcific InstanceVariable. In this example the 7th variable in the 2nd dataDscr element of the codebook always maps to the same reusable InstanceVariable.
AbstractSubstitution	Some references in the DDI4 model are to abstract classes. In these cases it is necessary to specify which extension of the abstract class should be used in the mapping.	Capture:RepresentedQuestion, DynamicTextContent:LiteralText	In this example a sourceCapture associates with the abstract class Capture. The mapping will use the RepresentedQuestion extension of the Capture.
ParameterValues	Some values in DDI4 can use additional explanatory metadata. This column lists the path and the value for that information.	RepresentedQuestion/hasInstruction/ InstructionText/textContent/purpose/ languageSpecificStructuredString/ content='PreQuestionText'	The LiteralText above is further described as "PreQuestionText"
IsIdRef	Is this value a reference to an ID in the DDI2 XML (an xs:IDREF)?, if so this will ultimately need to be transformed into a proper reference in DDI4 through a DDI URN.	TRUE	An example is the @var attribute of the DDI2 varGrp. This will need to be implemented as a reference to an InstanceVariable in a VariableCollection in DDI4.
DDI4IdRefPath	This is the sub-path within the last DDI4 identifiable object for the DDI URN of the referenced object.	contains/member	In the case of a varGrp, the VariableCollection has a contains/member value that is the URN of the DDI4 InstanceVariable created to match the DDI2 var referenced by the @var IDREF.
DDI4IdRefAppend	This describes whether to append or replace values in the DDI4 path.	TRUE, FALSE	In the case above, there may be more than one member property under contains, but there can only be one value for member.
Notes	Used to describe any notes for the mapping	requires matching DDI2 ID with DDI4 DdiUrn and inserting DDI4URN references into the VariableCollection	In this example, it describes what is needed in the matching process of DDI2 IDs and DDI4 URNS.

DDI25, DDI4PropertyName, and IdentifiablesMapping columns from DDI2 anlysUnit
AbstractSubstitution and ParameterValues from DDI2 preQTxt
IsIdRef, DDI4IdRefPath, DDI4IdRefAppend, and Notes from DDI2 varGrp/@var

D. DDI - CDI Upper Model / Dublin Core Map

Upper Model	Dublin Core	Notes
about	subject	dc is skos:broader
abstract	abstract	dc is skos:exactMatch
accessMode	format, type, medium	DC has a controlled vocabulary for type which includes the set of media types specified by the Internet Assigned Numbers Authority dc is skos:closeMatch
accessModeSufficient		
accessibilityAPI		
accessibilityControl		
accessibilityFeature		
accessibilityHazard		
accessbilitySummary		
accessRights	rights	dc is skos:exactMatch
accountablePerson	publisher	dc is skos:closeMatch
accrualMethod	accrualMethod	dc is skos:exactMatch
accrualPeriodicity	accrualPeriodicity	dc is skos:exactMatch
accrualPolicy	accrualPolicy	dc is skos:exactMatch
aggregateRating		
alternativeHeadline	alternative	dc is skos:exactMatch
associatedMedia	type	dc is skos:closeMatch
audience	audience	dc is skos:exactMatch
audio	type	dc is skos:broader
author	author	dc is skos:exactMatch
award		
character		
citation	bibliographicCitation	dc is skos:closeMatch
citationRequirement		
comment		
commentCount		
concept		

Upper Model	Dublin Core	Notes
conditionsOfAccess	available, accessRights, isRequiredBy, requires	dc is skos:narrower
confidentiality		
contentLocation	spatial	dc is skos:broader
contentRating		
contentReferenceTime	temporal, valid	dc is skos:narrower
contributor	contributor	dc is skos:exactMatch
copyrightHolder	rightsHolder	dc is skos:exactMatch
copyrightYear	dateCopyrighted	dc is skos:exactMatch
correction	provenance	dc is skos:broader
creativeWorkStatus	valid	dc is skos:narrower
creator	creator	dc is skos:exactMatch
dataQualityMetrics	provenance	dc is skos:broader
date	date	dc is skos:exactMatch
dateAccepted	date	dc is skos:broader
dateCreated	date, dateCopyrighted, dateSubmitted	dc is skos:narrower
dateModified	modified	dc is skos:exactMatch
datePublished	issued	dc is skos:closeMatch
dateSubmitted	dateSubmitted	dc is skos:exactMatch
disclaimer		
discussionURL	hasPart	dc is skos:broader
editor	contributor	dc is skos:broader
educationalAlignment	educationLevel	dc is skos:broader
educationalUse		
encoding	medium, type	dc is skos:closeMatch
encodingFormat	medium, type	dc is skos:broader
estimateOfSamplingError		
exampleOfWork	references, isReferencedBy, provenance	dc is skos:narrower

Upper Model	Dublin Core	Notes
expires	valid	dc is skos:narrower
funder	contributor	dc is skos:broader
genre	type	dc is skos:closeMatch
hasPart	hasPart	dc is skos:exactMatch
headline	title	dc is skos:closeMatch
inLanguage	language	dc is skos:closeMatch
interactionStatistic	hasPart	dc is skos:broader
interactivityType	instructionalMethod	dc is skos:broader
isAccessibleForFree	license	dc is skos:broader
isBasedOn	provenance	dc is skos:closeMatch
isFamilyFriendly		
isPartOf	isPartOf	dc is skos:exactMatch
keyWords	subject	dc is skos:closeMatch
learningResourceType		
license	license	dc is skos:exactMatch
locationCreated	provenance	dc is skos:broader
mainEntity	subject	dc is skos:broader
material	medium	dc is skos:closeMatch
materialExtent	extent	dc is skos:broader
mentions	references	dc is skos:closeMatch
notesOnDataCollection	provenance	dc is skos:broader
offers		
position		
producer	publisher	dc is skos:closeMatch
provider	contributor	dc is skos:broader
publication	provenance	dc is skos:broader
publisher	publisher	dc is skos:exactMatch
publisherImprint		
publishingPrinciples		
recodingAndDerivation		
recordedAt	provenance	dc is skos:broader

Upper Model	Dublin Core	Notes
releasedEvent	provenance	dc is skos:broader
researchQuestion		
researchHypothesis		
responseRate		
review		
schemaVersion		
sdDatePublished		
sdIdentifier		
sdLicense		
sdPublisher		
sdVersion		
sdVersionNotes		
security		
sourceOrganization		
spatial	spatial	dc is skos:exactMatch
spatialCoverage	spatial	dc is skos:broader
sponsor	contributor	dc is skos:broader
studyType	type	dc is skos:broader
temporal	temporal	dc is skos:exactMatch
temporalCoverage	temporal	dc is skos:broader
text		
thumbnailURL	hasPart	dc is skos:broader
timeRequired		
translationOfWork		
translator	contributor	dc is skos:broader
typicalAgeRange		
unitOfAnalysis	type	dc is skos:broader
universe	type	dc is skos:broader
version	hasVersion, isVersionOf	dc is skos:closeMatch
video	hasPart	dc is skos:broader
weighting		

Upper Model	Dublin Core	Notes
workExample	isFormatOf	dc is skos:broader
workTranslation	isFormatOf	dc is skos:broader

DDI – Cross Domain Integration: Detailed Examples and Use Cases

Contents

I. Overview

This document provides a set of detailed examples from the DDI – Cross Domain Integration (DDI – CDI) Data Description and Process specifications. Some of the examples use the XML representation of these specifications. Other examples paint a picture of how the XML representations might be transformed to tell the same story with other standards and formats. The intent here is to create representations at a level of specificity that will provide guidance to users who want to exercise the DDI - CDI standard in realistic situations. The first four examples together tell a data story, and this story has a context:

The Karonga Health and Demographic Surveillance System (Karonga HDSS) in northern Malawi currently has a population of more than 42 000 individuals under continuous demographic surveillance since completion of a baseline census (2002-2004). The surveillance system collects data on vital events and migration for individuals and for households. It also provides data on cause-specific mortality obtained by verbal autopsy for all age groups, and estimates rates of disease for specific presentations via linkage to clinical facility data. The Karonga HDSS provides a structure for surveys of socioeconomic status, HIV-prevalence and incidence, sexual behavior, fertility intentions and a sampling frame for other studies, as well as evaluating the impact of interventions, such as antiretroviral therapy and vaccination programs. Uniquely, it relies on a network of village informants to report vital events and household moves, and furthermore is linked to an archive of biological samples and data from population surveys and other studies dating back three decades.

Here is the data story:

- It begins with a DDI - CDI description of the HDSS event history data model that is able to capture both demographic and health events
- It continues with a description of the data workflow that loads HDSS operational data stores and their entities into the HDSS event history data model
- It continues with a description of the metadata workflow that reformats and turns the HDSS event history data model into a schema.org Dataset description
- It ends with a schema.org Dataset description of the HDSS event history data model produced by the metadata workflow above

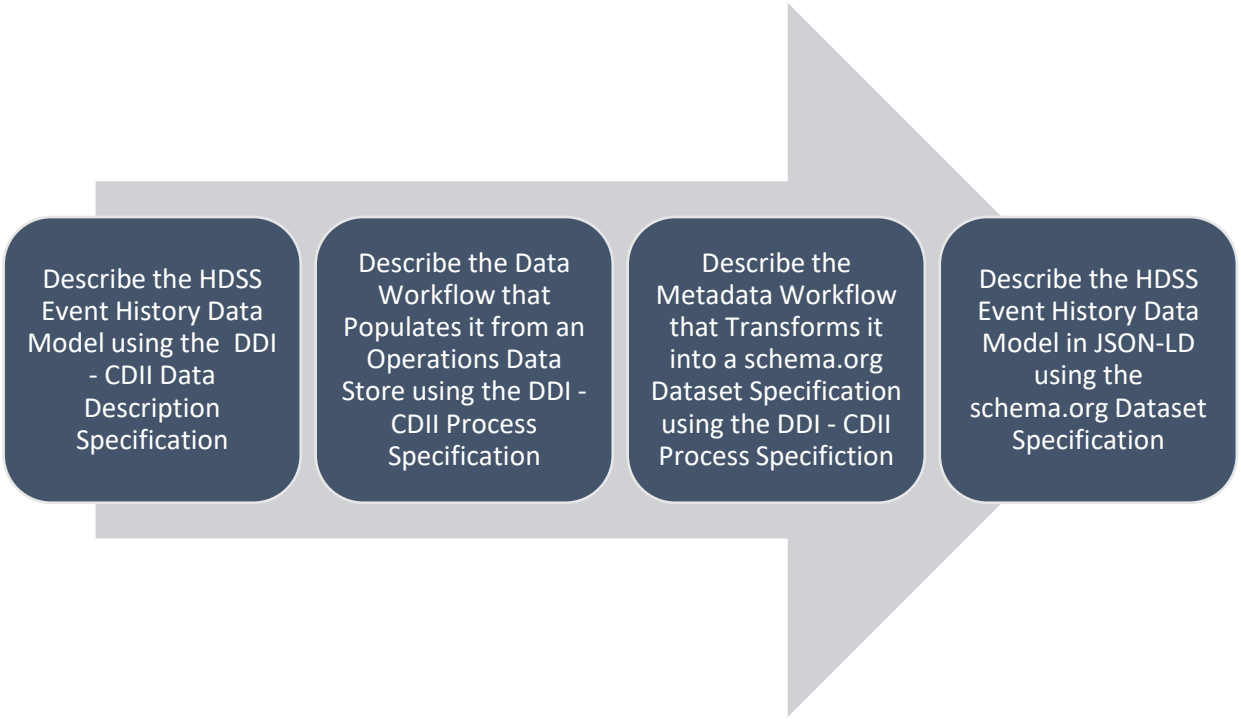


Figure 1: The Karonga HDSS Data Story Told in Four Examples

Additionally, there is an example of how a time series can be described using the DDI - CDI Data Description specification

II. The HDSS Event History Data Model Example

In DDI - CDI, following GSIM, a DataSet has a DataStructure, and the DataStructure has DataStructureComponents:

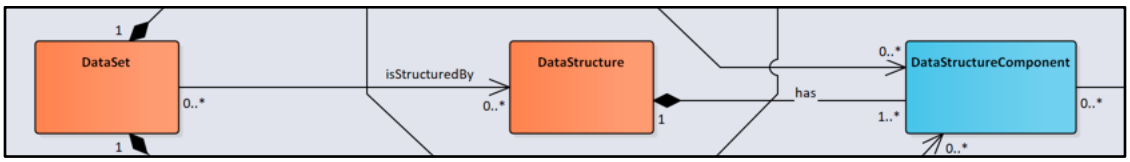


Figure 2: DDI - CDI DataSet Composition

DDI - CDI includes a rich set of components because it is intent on describing many types of DataSets that are encountered in research across many domains:

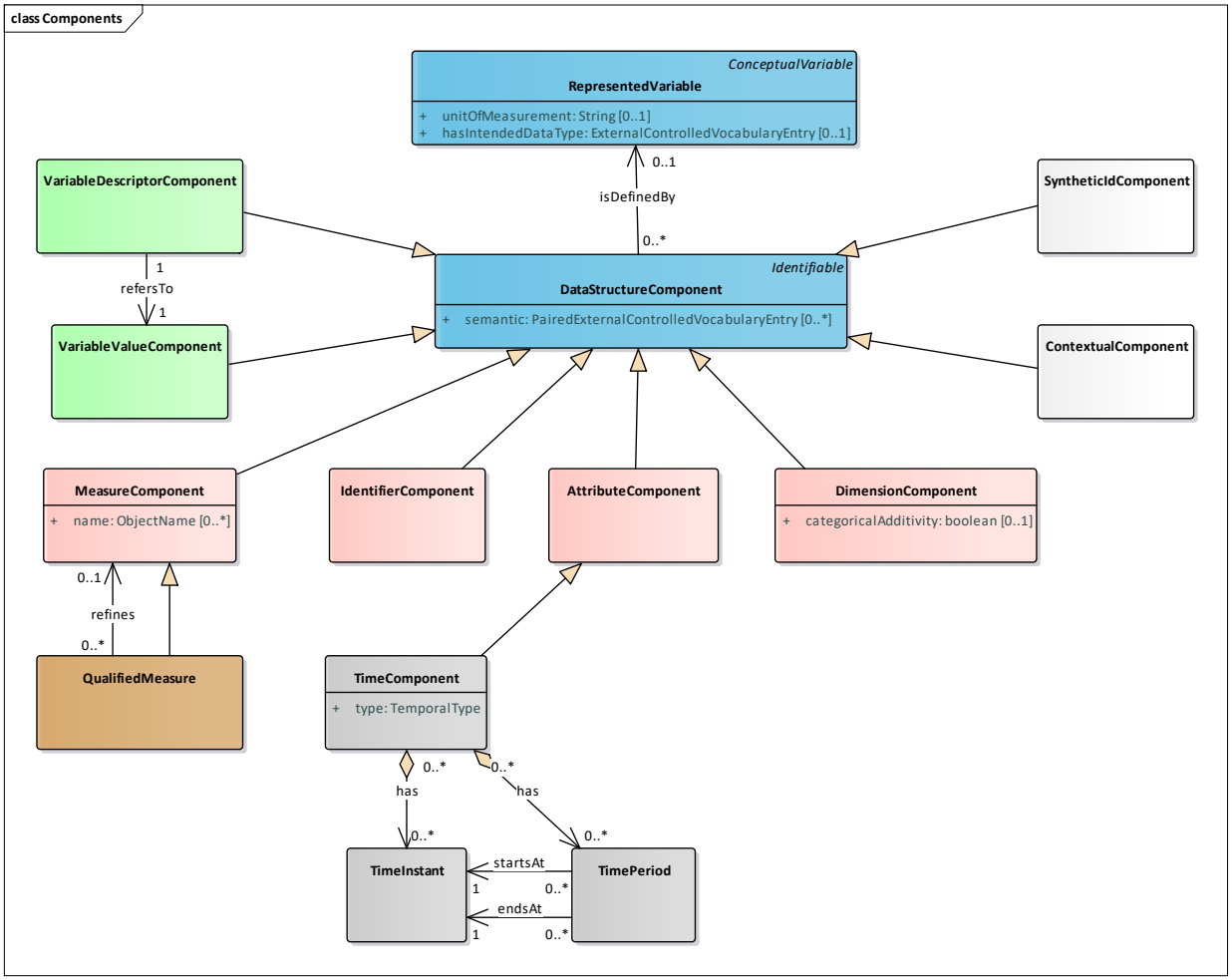


Figure 3: DDI - CDI DataStructure Components

In DDI - CDI Data Description the HDSS Event History Dataset has a LongDataStructure composed of a subset of the DataStructure components from Figure 3. With a LongDataStructure there is typically just one MeasureComponent per record[1]:

[1] There are two canonical variations of the LongDataSet and its LongDataStructure. In one variation, as happens with event histories, the measure does not change from one record to the next. In another variation, like what happens in the RAIRD Information Model, while each record has just one measure, that measure may change from one record to the next. In this variation the LongDataStructure, in place of a MeasureComponent, has a VariableDescriptorComponent and a VariableValueComponent. The VariableDescriptorComponent identifies a variable and the VariableValueComponent takes any value.

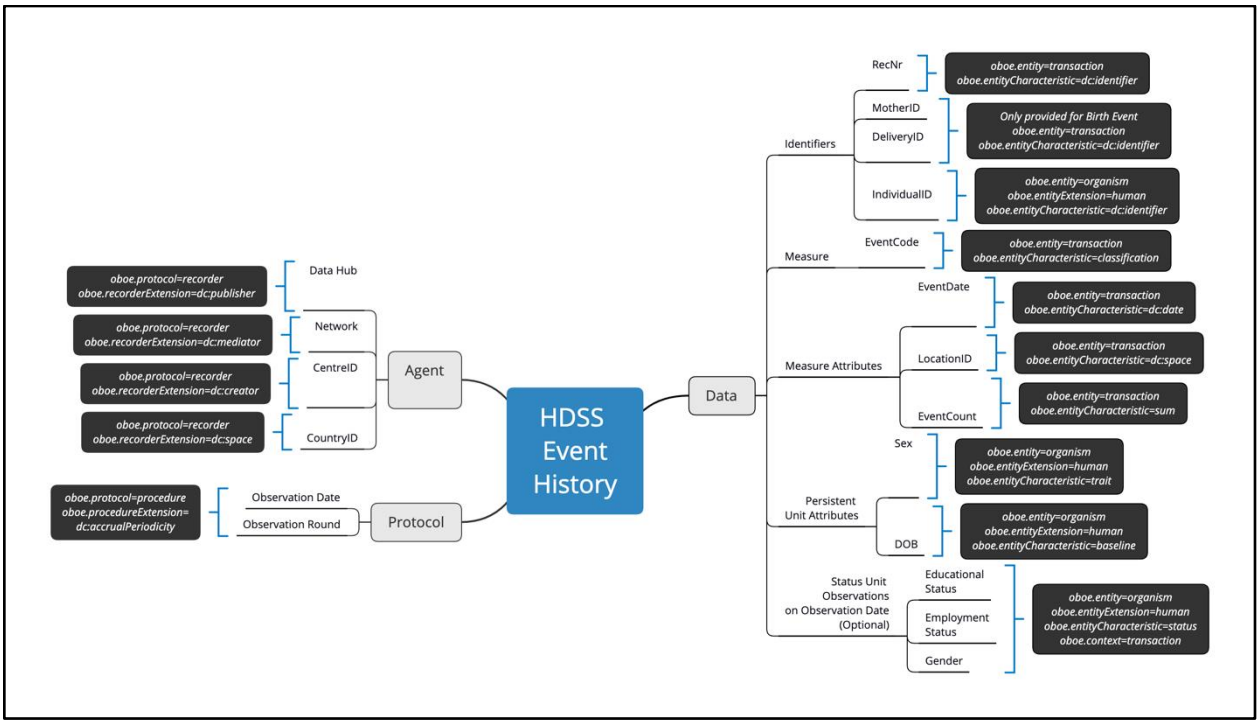


Figure 4: HDSS Event History DataSet DataStructure

Each observation in an HDSS Event History is an event with an event code; one or more event identifiers; some characteristics of the event including the time of the event and its location; some characteristics of the entity (unit) being observed that do not change from one event to the next in principle like sex and date of birth; some characteristics of the entity (unit) being observed that might change from the time of one observation to the next like educational status and employment status; and some characteristics of the protocol by way of which the event was observed like who observed the event and when the event was observed.

Surrounding the MeasureComponent in an HDSS Event History Dataset, then, there are one or more IdentiferComponents and a set of AttributeComponents specific to different entities that participate in the event history including the event itself, the person in general, the person at a point in time and the recorder/recording of the event.

As a consequence, in order to capture these characteristics, the IdentifierComponent(s), MeasureComponent and the AttributeComponents in an HDSS Event History are marked up semantically. DDI - CDI is indifferent to the markup language or, again, the ontology that is employed. Instead, in DDI - CDI, just like in other DDI products, there is a PairedExternalControlledVocabularyEntry which in DDI - CDI has been associated with a DataStructureComponent:

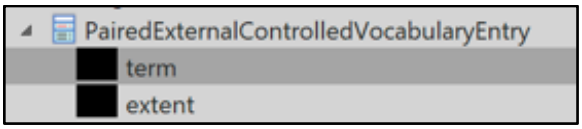


Figure 5: A PairedExternalControlledVocabularyEntry

Both the term and the extent in the pair are each an ExternalControlledVocabularyEntry:

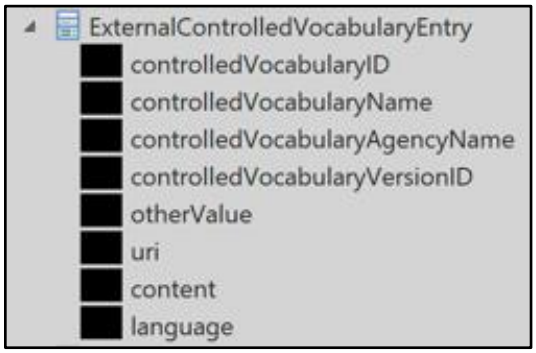


Figure 6: An ExternalControlledVocabularyEntry

The PairedExternalControlledVocabularyEntry is the data type of the semantic of all DataStructureComponents.

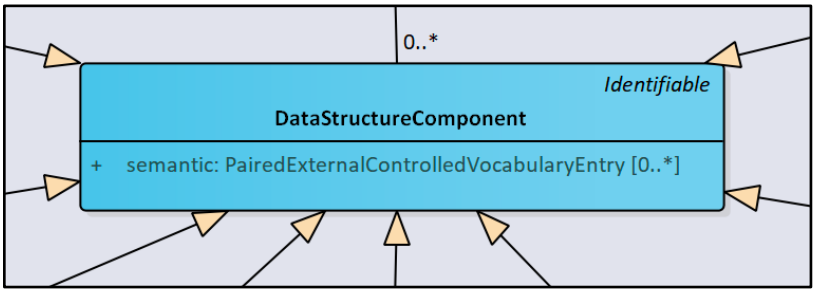


Figure 7: PairedExternalControlledVocabularyEntry is the data type of the semantic

The PairedExternalControlledVocabularyEntry is used by DDI - CDI to semantically mark up each DataStructureComponent. Each DataStructureComponent can take a succession of PairedExternalControlledVocabularyEntries in effect creating a controlled vocabulary with a hierarchical structure or, again, a taxonomy.[2]

[2] See Taxonomies and controlled vocabularies best practices for metadata by Heather Hedden for an in depth discussion of the use of taxonomies. Here she says: The word 'taxonomy' means the science of classifying things, and traditionally the classification of plants and animals, as in the Linnaean classification system. It has become a popular term now for any hierarchical classification or categorization system. Thus, a taxonomy is a controlled vocabulary in which all the terms belong to a single hierarchical structure and have parent/child or broader/narrower relationships to other terms. The structure is sometimes referred to as a 'tree'. The addition of non-preferred terms/synonyms may or may not be part of a taxonomy.

Although DDI - CDI is "indifferent" to the markup language selected, the one that is employed here with the HDSS Event History Dataset is called the Extensible Observation Ontology (OBOE). OBOE, simply put, describes entities being observed and their characteristics:

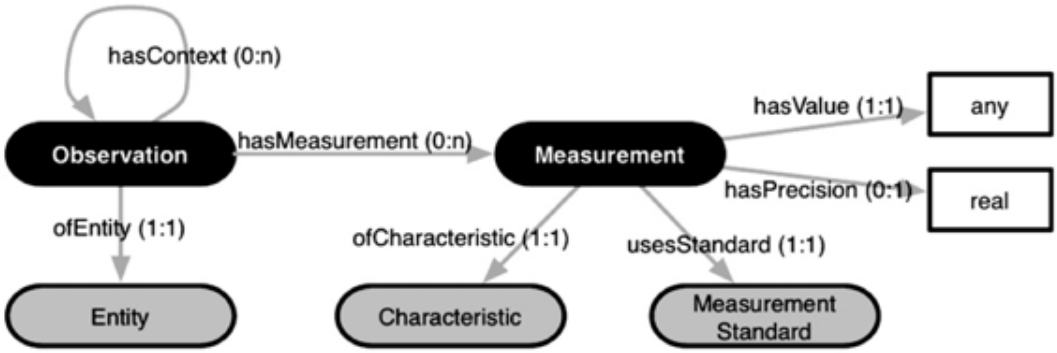


Figure 8: The core classes (ellipses) and properties (arrows) of the Extensible Observation Ontology (OBOE). Each Observation is of some Entity and can provide context for the Observation of another Entity. A Characteristic of an Entity can be represented through a Measurement. Measurements relate Characteristics to a Measurement Standard via a Value and, if applicable, a Precision. Measurements are taken by a Recorder (human or non-human) using a Protocol at a particular Time and Place

OBOE is a so-called "observation" ontology used in ecological research. OBOE was recently aligned with the Semantic Sensor Network Ontology. OBOE markup for the HDSS Event History Dataset appears in the black boxes of Figure 4 above.

An XML representation of the HDSS Event History Data Model used by many HDSSs across Sub-Saharan including Karonga can be found here[3].

III. An HDSS Data Workflow Example

In this example just a fragment of the actual HDSS data workflow is described. The fragment contains three Activities each of which contain several Steps:

[3] This example XML references two other files – DDI_CDI.xsd and xml.xsd. All three files need to be placed in the same directory.

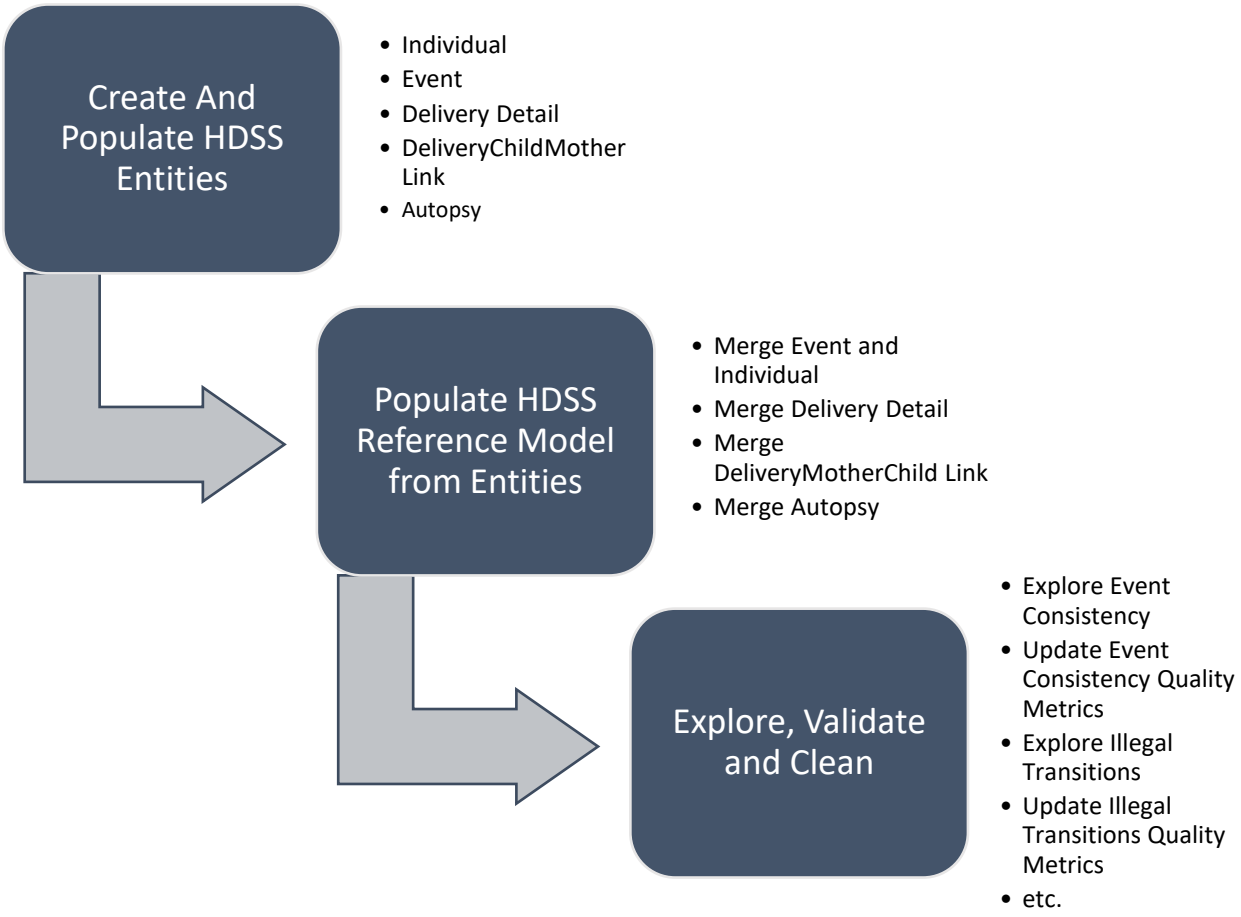


Figure 9: The HDSS Pentaho Data Workflow (Fragment)

The succession of Activities is sequential. Within the *Create and Populate HDSS Entities* Activity the succession of Tasks runs in parallel. Within both the *Populate HDSS Reference Model from Entities* and *Explore, Validate and Clean* Activity the succession of Tasks is sequential.

Finally, each Task uses and/or produces one or more InformationObjects with the execution of each of these Tasks conditioned by the existence of the InformationObject(s) it uses. In this example, parameters and the pathways they follow from one Task to the next are not utilized. Instead, Tasks communicate by checking the existence of InformationObjects in the data store. In fact, this is the approach HDSSs are taking in Pentaho. Had a different platform and/or approach been taken, the description might have utilized parameters and pathways. Both approaches are supported in the DDI Process specification.

A DDI - CDI XML representation of this data workflow can be found here.

IV. A Metadata Workflow Example

This workflow is a future. It's idea, however, grows out of an actual product that was built by members of the DDI community – DDI2R. DDI2R constructs an R class library based on a DDI - CDI profile. Along the way it has been determined that perhaps the "right" architectural solution for DDI2R is to build an

intermediate class library in Python which might be used to construct a family of products including DDI2R. The workflow below is based on that idea:

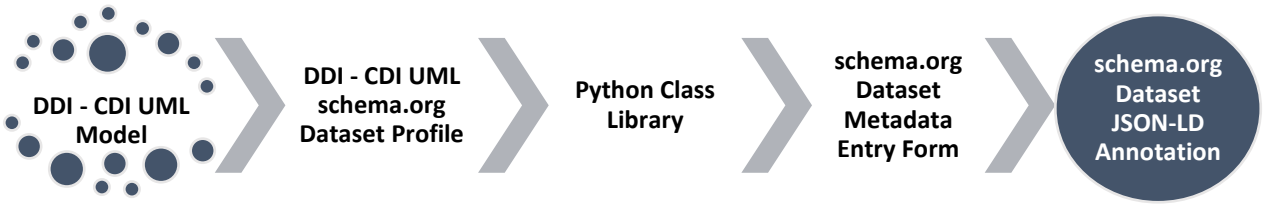


Figure 10: DDI2JSON-LD

DDI2JSON-LD constructs a schema.org Dataset annotation organizations might use to make the datasets they publish on the web discoverable by Google Dataset Search.

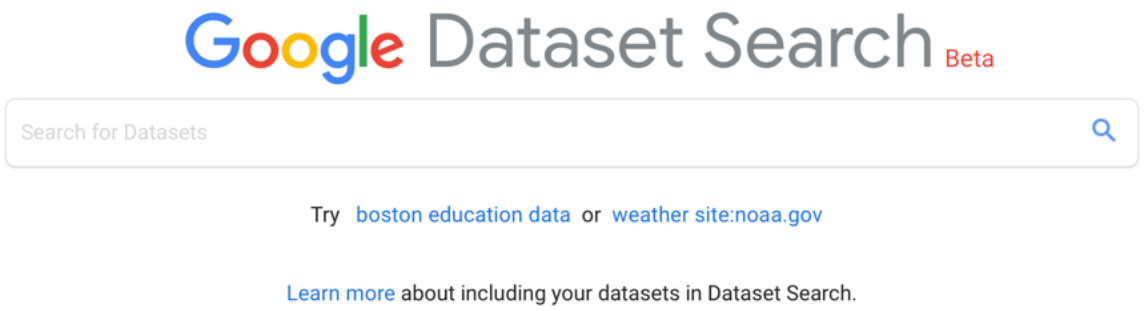


Figure 11: Google Dataset Search

Recall that in the recent past Google launched Dataset Search "so that scientists, data journalists, data geeks, or anyone else can find the data required for their work and their stories, or simply to satisfy their intellectual curiosity." According to Google, "Dataset Search lets you find datasets wherever they're hosted, whether it's a publisher's site, a digital library, or an author's personal web page."

In this workflow a schema.org Dataset metadata entry form is created which an organization can complete. From the completed form a JSON-LD annotation is produced that the organization can attach to each dataset the organization publishes on the web.

The form is notionally built using the Python Class Library which is, in turn, based on a DDI - CDI schema.org Dataset profile. That profile includes both variable-level objects from the DDI - CDI Data Description specification and objects from the so-called Upper Model. The Upper Model is not currently in scope for DDI - CDI. It provides the research context. A discussion of the research context and the Upper Model can be found in the Section IIC1 of the Architecture Document.

The actual workflow depicted in Figure 9 is not very big. It has been rendered here in XML using the DDI - CDI process model.

V. The Karonga HDSS schema.org Dataset

In fact, the construction of this workflow that the Karonga HDSS might use to produce annotations for the datasets it publishes is a work in progress.

In the interim a generic annotation for the HDSS Event History Reference Model used by many HDSSs including Karonga to conduct demographic and epidemiological was constructed by hand. It can be found here.

And here are a few illustrative schema.org Dataset variable descriptions taken from that annotation. They come from the Google Structured Data Testing Tool. This tool takes as input schema.org JSON-LD and produces as output a pretty description of the JSON-LD like a user would see during Google Dataset Search.

The illustrative variable descriptions are each annotated to facilitate a short discussion that follows each snapshot.

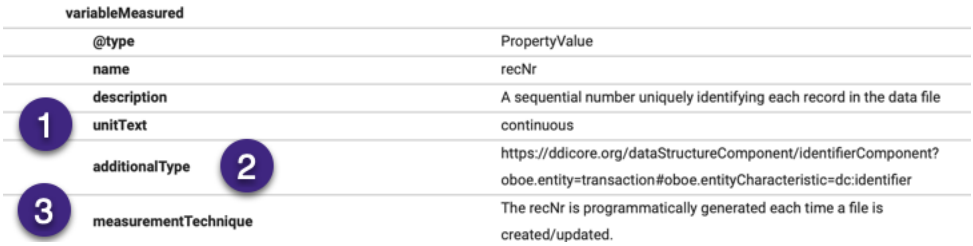


Figure 12: The recNr variableMeasured from the HDSS Event History Dataset is a "sequential number uniquely identifying each record in the data file". (1) unitText is a string or text indicating the unit of measurement. It is useful if you cannot provide a standard unit code for unitCode (from schema.org). (2) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – an IdentifierComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). The OBOE entity measured by recNr is a transaction. And the entity characteristic measured by recNr is a Dublin Core identifier. (3) measurementTechnique is not always included. Here it is integral to the variableMeasured.

variableMeasured	
@type	PropertyValue
name	eventCode
description	A code identifying the type of event that has occurred
1 unitText	discrete
additionalType 2	https://ddicore.org/dataStructureComponent/measureComponent?oboe.entity=transaction#oboe.entityCharacteristic=classification
3 propertyID	https://hdss.org/eventType
identifier	
@type 4	PropertyValue
url	https://hdss.org/eventType/enumeration/ENU
description	Starting event for all individuals present at the baseline census of the surveillance area. It corresponds to the date on which the individual was first observed to be present in the surveillance area during the baseline census.
identifier	
@type 4	PropertyValue
url	https://hdss.org/eventType/birth/BTH
description	The birth of an individual to a resident female.
identifier	
@type 4	PropertyValue
url	https://hdss.org/eventType/inmigration/IMG
description	The event of migrating into the surveillance area.
identifier	
@type 4	PropertyValue
url	https://hdss.org/eventType/outmigration/OMG
description	The event of migrating out of the surveillance Destination (same as for Origin) area.

Figure 13: The eventCode variableMeasured from the HDSS Event History Dataset is a "code identifying the type of event that has occurred". (1) The unit of measurement is discrete. (2) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – a MeasureComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). The OBOE entity measured by eventCode is a transaction. And the entity characteristic measured by eventCode is a classification. (3) The data type is an HDSS codelist called eventType. (4) is a partial enumeration of this codelist in which each entry consists of a label and a code.

variableMeasured	
@type	PropertyValue
name	observationDate
description	Date on which the event was observed (recorded), also known as surveillance visit date
1 unitText	discrete
propertyID 2	https://java.com/localdate
3 additionalType	https://ddicore.org/dataStructureComponent/attributeComponent?oboe.protocol=procedure#oboe.procedureExtension=dc:accrualPeriodicity

Figure 14: The observationDate variableMeasured from the HDSS Event History Dataset is a "date on which the event was observed (recorded), also known as surveillance visit date". (1) The unit of measurement is discrete. (2) The data type is a Java localdate. (3) additionalType here both provides the DDI - CDI DataStructureComponentType of the variableMeasured – an AttributeComponent – and its semantic markup using the Extensible Observation Ontology (OBOE). In OBOE observationDate belongs to the protocol. Within the protocol it is a procedure. More specifically it represents Dublin Core accrualPeriodicity.

VI. Cell-Oriented Time Series Example

DDI - CDI supports the descriptions of time series as part of a general multi-dimensional structure, where time is the dimension used to connect observations, as one among many. This is an OLAP-based approach to time series, but there are many systems in use today which do not handle time series in this fashion. This example uses a "cell-based" approach which is more appropriate for these systems – it

demonstrates the flexibility of DDI - CDI when it comes to describing the data needed to support the needs of a particular implementation.

In this section, we provide an example for how to describe a time series. We will use the Urban Consumer Price Index (CPI-U) from the US Bureau of Labor Statistics.

CPI-U is a family of indexes, each available as a series going back many years. Some go back to 1913. Each index has a base year from which the current value is derived. The base years are mostly in the range 1982-1984. The value for the base year and period is set as 100.0.

Series data in general are dimensional. Unlike a cube where every dimensional combination (i.e., the combination of categories taken from each one of the dimensions) is represented, in a series we are interested in looking at one dimensional combination over time. One may conceive of this as looking at one cell in a cube taken over time. For the CPI-U, the dimensions available are Item (product or service) and Area (metropolitan statistical area). Each series in the CPI-U family represents the combination of one Item category and one Area category. It is possible to ignore one or the other dimension by selecting "all" instead of a specific entry in each dimension. Selecting the "all" category effectively collapses the dimension.

The CPI-U is published as an overall index for the entire urban US, for individual items (products and services), urban areas, and combinations of items and areas. Some series in the family are published as either seasonally adjusted or not. We will illustrate a non-seasonally adjusted time series for the CPI-U for Apparel in Washington, DC. Note, a seasonally adjusted series is a conceptually different series than a non-seasonally adjusted one. This adjustment is not a kind of revision for each number in the series. Seasonal adjustment affects the entire series.

The lists of items and areas, the Dimensions, are Code Lists as defined in DDI. Here are short illustrations of the Items and Areas dimension files:

Area

Code	Area
35A	Washington-Arlington-Alexandria, DC-VA
S35B	Miami-Fort Lauderdale-West Palm Beach, FL
S35C	Atlanta-Sandy Springs-Roswell, GA
S35D	Tampa-St. Petersburg-Clearwater, FL
S35E	Baltimore-Columbia-Towson, MD
S37A	Dallas-Fort Worth-Arlington, TX

Item

Code	Item
SA311	Apparel less footwear
SAA	Apparel
SAA1	Men's and boys' apparel
SAA2	Women's and girls' apparel
SAC	Commodities
SACE	Energy commodities

The codes and categories are listed in the order they appear in the BLS files.

The table below contains CPI-U for Apparel (code – SAA) in Washington, DC (code – 35A) as a non-seasonally adjusted series going back 10 years for a bi-monthly period:

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2009	138.785		146.027		144.305		142.143		147.452		140.421	
2010	136.404		147.671		140.853		134.526		144.549		138.684	
2011	142.773		151.650		154.093		151.442		151.071		153.376	
2012	143.069		159.347		155.926		144.799		160.952		153.936	
2013	139.126		142.290		144.985		140.668		151.600		149.854	
2014	141.130		147.795		148.749		139.189		156.178		148.931	
2015	135.237		153.824		148.202		134.230		147.512		141.474	
2016	142.103		153.600		159.872		151.601		162.246		152.816	
2017	152.014		153.619		157.312		149.154		165.510		154.720	
2018	164.464		162.120		163.558		156.946		177.968		165.956	
2019	169.674		167.026		170.495		157.230					

The series includes the most recent estimate available at the time of the drafting of this document. Note also, the frequency of the estimates in this series is bi-monthly. Even though the CPI-U for all items and all areas (the US estimate) is issued every month, the data here do not support a monthly release for this index. Some other detailed indexes in this family are released every month.

Another issue about indexes is they cannot be compared across series. For example, the index for New York City for Apparel in July 2019 is 116.924 and that for Washington, DC is 157.230. This does not mean, however, that apparel is more expensive in Washington than in New York. Indexes are relative to the item and area they represent.

Technical Committee

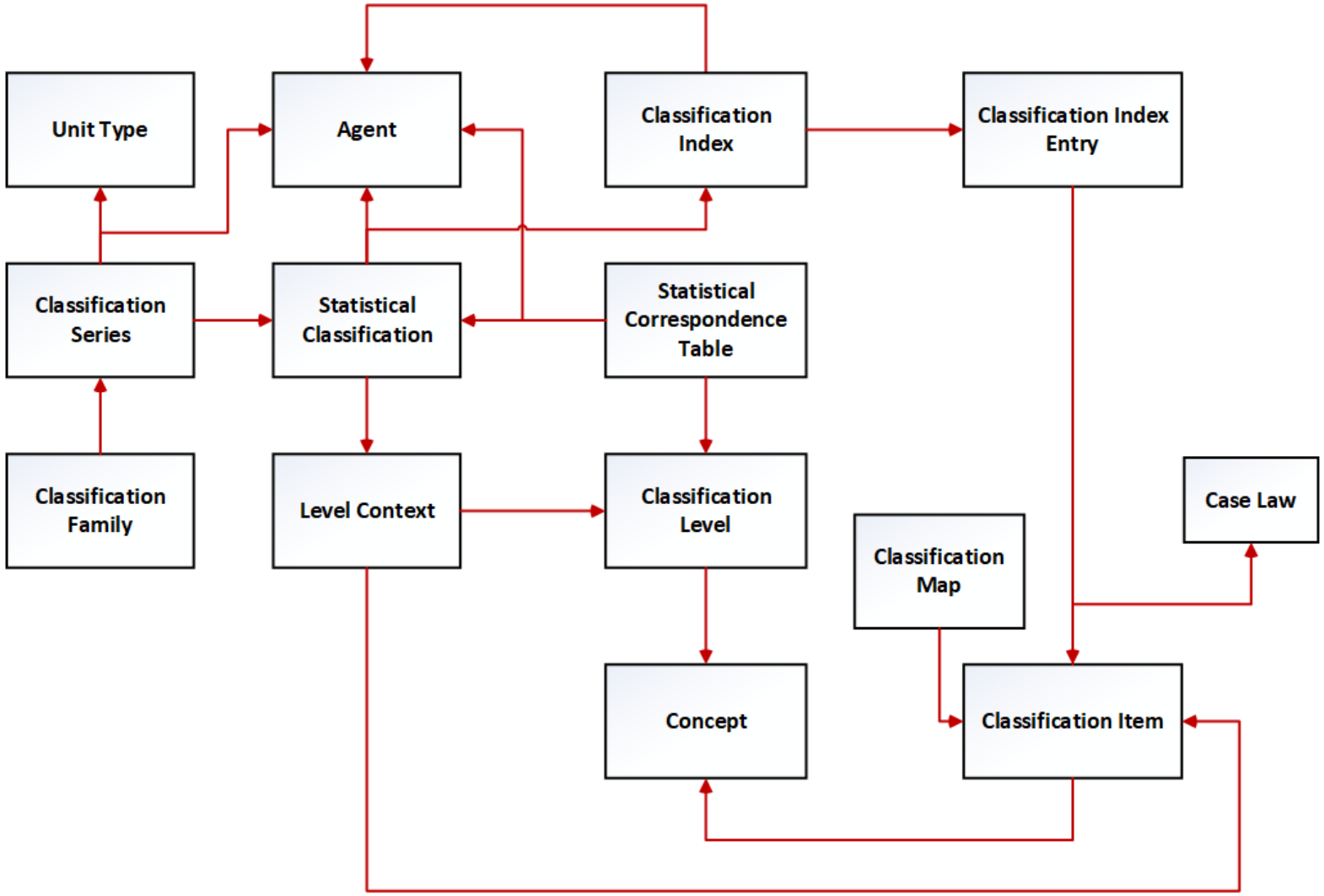
Wendy Thomas and Jon Johnson

18 May 2020

DDI Lifecycle 3.3 – New Features

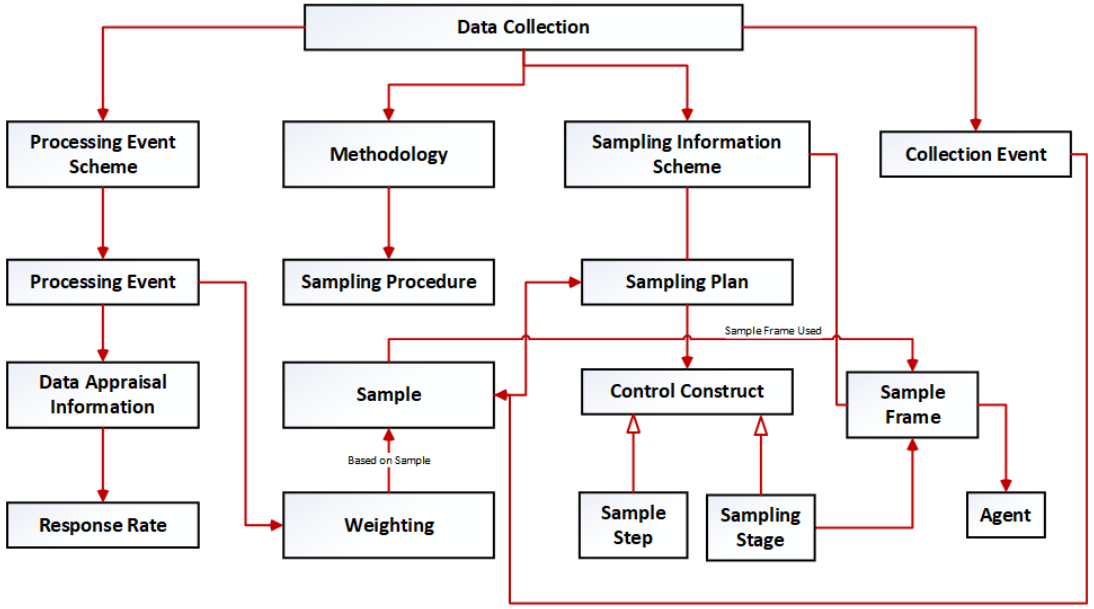
- Statistical Classification
- SDI work (Data Capture Development, Sampling, Weighting)
- Measurement items
- Quality and certifications
- Support for Property Graphs
- Conceptual objects updated
- Expanded documentation

Statistical Classification

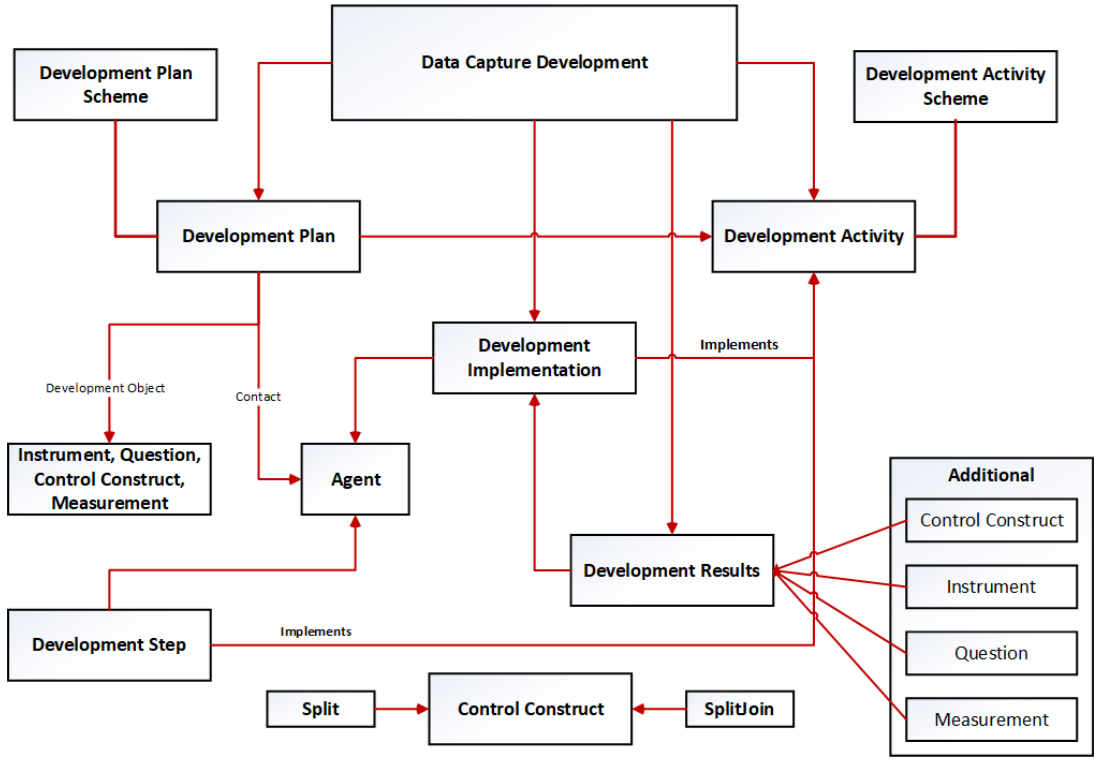


Survey Development & Implementation work

Sampling



Data Capture Development



Expansion of Lifecycle – Version 3.4

- Purpose of version 3.4 is to test out new style of XML plus expressing DDI-Lifecycle in additional bindings
- Content will be consistent with 3.3 – minus some of the organizational packaging (Data Collection, Logical Product, Conceptual Content, Comparison, Archive, Physical Product, Physical Instance for example)
- Simplification of Identifiable, Versionable, and Maintainable approach to identification
- Finalize move to an automated production system that will support iterative testing of new content and generation of multiple bindings
- WE NEED YOU! We need to get requirements and suggestions for each binding in terms of binding type and style

Codebook

- Short term work
  - Specific issues – needed content (new elements and attributes)
  - Support for management systems – is there additional content that can help implementers
  - Interaction with other metadata systems – clarification of content, new content, other identified issues
- Long term work
  - Description of what Codebook needs to be in terms of coverage and application (what niche does it fill)
  - What are the implications of clearly declaring Codebook a product within a suite of DDI products on development rules

Roadmap – Defining the DDI Suite of Products

- Describing DDI as a suite of products
- Describe individual products in terms of their coverage within the full suite, technical requirements, intended applications, and binding/representation options
- Use the overall and specific models to explore alignment with external standards
- Use models to facilitate the translation of metadata between products
- Explore the creation of an overarching conceptual content model to aid in describing DDI content coverage overall

Call for new TC members

- TC is a standing committee of the Scientific Board. Its role is to:
  - Provide and oversee the technical production of DDI products
  - Provide adequate coverage of content areas identified by the Scientific Board as belonging to DDI
  - Ensure continued maintenance of current and future products
  - Ensure DDI products meet the technical requirements of users over time
  - Address technically how the products of DDI interrelate and work together
- Our members have an overriding interest in these broad issues and focus on one or more specific areas of work (overall, documentation, specific products)
- Contributing to the work of the Technical Committee
  - Join the TC as a member
  - Contribute to specific tasks of the Technical Committee in the coming year

Background information

- Technical Committee Wiki Page
- Technical Committee Annual Report
- October 2019 meeting outcomes
- Workplan for 2020/21

Technical Committee Report for 2019/2020

Submitted by: Wendy Thomas, Chair on behalf of the Technical Committee

Work completed in 2019/2020

The Technical Committee focused most of this past year on 2 major areas, product publication and shifting our production work to an automated system in line with the DDI Roadmap work.

- The primary accomplishment of the Technical Committee over the past year has been the final review, modification, and publication of DDI-Lifecycle Version 3.3. During this process we were able to test out the documentation production features of the COGS system, resulting in refinements and adjustments in our approach. The new version of Lifecycle covers a significant expansion of the standard into areas where it has been week
    - Opened up existing parts of the standard to broader application in the areas of data capture (expanding to non-question-based capture) and flexibility in the use of the control constructs (Lifecycle process model) for describing data management processes
    - Improved relationship to the work of GSIM including an improved alignment of conceptual content and the addition of Statistical Classification in line with the GSIM model
    - Added the content developed by the Survey Development and Implementation working group covering:
        - Data Capture Development – The development and testing work associated with creating and fielding data capture instruments including question and measurement development, translation, quality testing, and delivery methods
        - Sampling – The methodology used for sampling, management of sample frames, and defining complex sampling processes
        - Weighting – The source, process, purpose, and guidance for weighting. This includes a means of instructing analysis tools on the appropriate use of weights in analysis.
    - Revision of Methodology, Quality Information, and Variable Cascade content to increase clarification on the purpose and use of this content.
    - Support for DDI as a Property Graph (properties on items and references)
    - Quality Statement improvements (useful for Eurostat reporting)
- Thanks to the continued work of Franck Cotton and Thomas Francart the new product XKOS – Extended Knowledge Organization System was published in June 2019. XKOS is currently under use by INSEE, FAO, and the UN as a means of publishing Statistical Classifications. XKOS is currently accepting and resolving issues with a small group of dedicated members.
- In October 2019 the TC held a face-to-face meeting funded by the Executive Board. Meetings in the previous 3 years had been small targeted meetings with 3-4 members dealing with a limited number of issues. The 2019 meeting, hosted by the Minnesota Population Center, had 3 main work areas: DDI Roadmap, Production Framework, and DDI-Lifecycle Version 3.3 review and entry. This meeting supported the completion of a good deal of work in a short, concentrated period. A full report is available at https://ddi-

alliance.atlassian.net/wiki/spaces/DDI4/pages/723255303/In-person+Meeting+21-25+October+2019?preview=/723255303/731054081/TC_October_2019_Meeting_Summary.pdf

- The TC worked with the MRT on setting up for the public review of DDI-CDI Cross Domain Integration currently underway.
- We continue to work with the Controlled Vocabulary group in identifying and publishing new and updated Controlled Vocabularies. We have increased documentation in DDI-Lifecyle to strengthen the tie between DDI Controlled Vocabularies and their use in DDI-Lifecycle.
- The TC has begun discussions of the implications of the DDI Roadmap for organizing, describing, marketing, and training for DDI products.

Workplan for 2020/2021

The Technical Committee is exploring how we do our work this year. In the past few years we have viewed our membership as individuals who have both an overall interest in the technical products of the DDI and specific areas of concern. We have been flexible in terms of meeting attendance based on agenda items and on encouraging feedback to the group through comments on JIRA issues, wiki content and email communications. In taking this approach we try to keep the burden on TC members reasonable, acknowledging not all members have interests in all areas we work in. This year we are exploring how to bring in people from the DDI community to work on specific task areas. We have recently started with a call for input to a DDI Codebook review and update. The immediate response was very good, and we continue to invite people into this process. We have also identified small working groups within the TC who are focusing on specific goals for the TC, using the regular TC meeting time to support the work of these smaller groups when possible. The major task areas for this year are:

- Continued improvement of the DDI-Lifecycle high level documentation. With the publication of DDI-Lifecycle version 3.3 this type of documentation has been separated from the specification package allowing updates to the content without having to version the specification. Some goals of this work
    - Work with Training Working Group to provide content that can be reused in training products, such as images and graphics, which should be consistent between the two platforms
    - Republish high level documentation for DDI-Lifecycle version 3.2 with material common between versions 3.2 and 3.3
    - Explore a set of documentation that supports implementation of DDI-Lifecycle on a technical level
- DDI-Lifecycle version 3.4 – This next version has specific goals
    - The content will be the same as version 3.3 but the expressed in a technical structure that supports more flexibility
    - Express content in multiple bindings/representations: XML Schema, RDF/OWL, JSON, and UML
    - Move to an automated production process that will expand the ability to support and test new content and work in a more iterative manner
    - Test the transformation processes for input and output to this system to ensure lossless transfer of content into the internal registry and export of consistent content to the various bindings/representations

- DDI-Codebook review and update
    - We have recently put out a call for those interested in working on an updated version of DDI-Codebook to be reviewed and published this year. We are soliciting additional issues and looking for individuals who would like to assist the TC with this work. Initially we will focus on issues to be resolved for the update. These include suggestions to improve interaction with DataVerse, support for new projects using or interacting with DDI-Codebook and covering some related content that has been added DDI-Lifecycle. The goal is to improve functioning within the current DDI-Codebook development constraints.
    - Long term we will be discussing the role of DDI-Codebook within the DDI Suite of Products and any implications for future coverage, structure, and development rules.
- Roadmap work – Definition of the DDI Suite of Products and each product produced by the DDI Alliance
    - Creation of an overarching conceptual content model for the DDI Suite of Products
    - Describe individual products in terms of their coverage within the full suite, technical requirements, intended applications, and binding/representation options. We will be working with members of the different product groups as well as Marketing and Training to development this information and present it in appropriate ways for different purposes.
    - Use the overall and specific models to explore alignment with external standards to identify differences in coverage, perspectives, gaps that should be addressed, and points where metadata and data need to transfer between systems.
    - Use models to facilitate the translation of metadata between products. This will help organizations managing DDI content who need to interact with users who require different products for different applications.

Technical Committee Meeting 21-25 October 2019: Summary

Attendees: Wendy Thomas, Jon Johnson, Larry Hoyle, Jeremy Iverson, Dan Smith, Johan Fihn Marberg, Oliver Hopt, Dan Gillman (as available), Barry Radler (Monday-Tuesday), Flavio Rizzolo (remote, as available), Jay Greenfield (remote, as available)

Meeting Page:

https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/723255303/In-person+Meeting+21-25+October+2019

The Monday/Tuesday agenda was adjusted to fit the availability of remote members. Entry and review of DDI 3.3 changes were added to the agenda as well as review of outstanding TC issues.

Results of Meeting:

DDI Roadmap:

This discussion took place within the context of current and future user groups of DDI products, the mix of DDI products, marketing, production frameworks, and integration. The initial point of discussion was the meaning of a single "DDI". Given the expansion of DDI products outside of the "DDI specification" as well as the technical constraints and content needs expressed by various user groups the goal of a single integrated DDI specification seemed difficult to attain. Instead, the idea of DDI as a suite of products which covered a specified topical and application area unified by a common conceptual model emerged. In this scenario each product would specify coverage of the conceptual model in terms of content and application support. Development efforts would be towards improving consistency across the products to support transfer of content from one product to another, clear mapping content between products, and a shift in the definition of the DDI products to focus on their content coverage and applied uses. From a marketing perspective it allows for promoting DDI as a whole while directing potential users to specific products in the suite that support specific activities and technical environments.

Notes from this discussion are found in the TC Minutes under 20191021 – 20191025

TASK: Complete draft of proposal for updated DDI Roadmap for member review through the Scientific Board

Production Framework

New TC Production Framework page listing information for DDI-Lifecycle and Controlled Vocabularies Pipeline

https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/729284679/TC+Production+Framework

COGS

COGS is a development and production environment that takes a "registry" approach to the management of DDI content. Identified-Versioned objects are intended to be managed by registries for the purpose of reuse. The content of DDI may be expressed in XML, RDF, XMI, JSON, as well as other commonly used expressions. Documentation and graphs of objects are also produced. By managing the content of these objects using CSV files in a GIT repository, a broad range of implementers and users can develop content, test it in an iterative manner, and submit it for inclusion in DDI. The use of COGS supports:

- Automated verification of content against design rules
- Automated production of multiple expressions of the DDI content to support various uses and applications
- Options for iterative testing of new content
- Options for increased involvement in DDI development by lowering the technical bar for participation

The applicability of COGS to the main DDI products was discussed. Initial usage was envisioned for DDI-Lifecycle. Discussions of the utility of COGS for DDI4 was focused on the production of documentation. There is a difference in perspective between the UML modeling approach and the COGS Registry approach. Specific issues were identified and these should be the basis for further discussion on the maintenance of DDI4 over time. One point affecting the XMI output of COGS (currently EA flavor of normative OMG UML 2.4.2 and 2.5 with diagrams) is that of the appearance of diagrams and their content. XMI output of COGS should meet the expectations and needs of the UML user community. The possibilities for the application of COGS to DDI-Codebook were discussed and will be examined over the coming year. The move of either the DDI-Lifecycle or DDI-Codebook specifications would result in a structurally different XML output. Some design rule changes such as dropping the use of global attributes or unions, and the substitution group approach for Physical Structure would drive these changes. The proposal is to take DDI-Lifecycle 3.3, move it to COGS, and produce a DDI-Lifecycle 3.4 for technical review so that developers can focus on the structural and design changes independent of content additions. Dan Smith will do a test load of DDI-L 3.3 when last few items are entered. A similar approach would be used to examine a change in the DDI-Codebook schema.

Documentation Production

Work was completed on resolving publication and production issues for the DocFlex production of HTML documentation for DDI-Lifecycle and DDI-Codebook object level metadata.

Production of the high-level documentation through COGS based on restructured text files is now functional and only requires additional content coverage prior to publication. This is an on-going project and was not part of the October meeting agenda. Object level documentation from COGS will be not be provided as part of the publication package for DDI-Lifecycle 3.3 as it reflects the design rules that will be reviewed by DDI-Lifecycle 3.4.

There was discussion of what documentation should be included in the official publication package and what should be supplemental. Separating high-level documentation as well as alternate distribution structures allows some flexibility for improving documentation between official publications.

Controlled Vocabulary Production – post CESSDA

Clarified the output content of the CESSDA CV management system. All outputs (HTML, SKOS, PDF) have multi-lingual content as required by DDI. The addition of a language selection bar on the HTML output allows for filtering by language. DDI would also like to provide a multi-lingual output of each CV in a common DDI-Lifecycle CodeList structure allowing users to point to a content structure that was already familiar to them. Oliver wrote the needed transformations and tools pipeline to produce a package of output for updating the ICPSR hosted site for Controlled Vocabularies. Use of this pipeline reduces the workload for updating based on new publications to updating of the page listing all current CV's and their download options and the page listing version histories. Currently CESSDA does not provide a

"push" option for initiating this process (it is planned for the future). A command line tool has been created to grab the CESSDA output, run it through a DDI production pipeline, and provide ICPSR staff with a zipped package for publication. TC is currently coordinating this work with the CV working group and Michael Iannaccona of ICPSR. See document on Controlled Vocabularies on the Meeting Page.

DDI Resolution

URL format for API resolution (resolving a DDI URN to an API format) – did not get to

TC-4 Informed that this is now progressing again – keep tracking

High Level Documentation Activities

Integration

If the approach of a DDI Suite of products is supported by the DDI Community this implies the need for a number of documents:

- Creation of a conceptual model underlying DDI – Flavio will explore this and we will add to future TC agenda
    - Conceptual level mapping of products to the overall model
    - Use to identify commonalities, points of similarity/dissimilarity
    - Use of this information for Marketing and Training to explain DDI overall and applications for specific products
- Explore implications for the design and development rules for each product in terms of coverage and technical implementation
    - How does this effect the addition of new content across products

Mapping

A first pass at mapping DDI4 Core content objects to DDI-Lifecycle 3.3 complex elements was started. A mapping page will be created to bring together this and other product-to-product mapping information. Progress was delayed until DDI4 specific (property and relationship level content) was solidified and the DDI-Lifecycle 3.3 updates were completed. Many of the mapping issues are found in the property and relationship details so that having stable content for mapping is a must. The approach for publication will start with a spreadsheet as an easy means for both visualization and processing of information. While many areas have relatively clear object to object mapping, changes regarding the logical and physical description of data are extensive and resulted in a many-to-many mapping at the object level. This section must be done property to property. This work must be continued post-meeting.

Best Practices Document

The Best Practices Document for DDI 3.2 and future versions was reviewed and updated. Dan Smith will enter updates and provide the PDF for version 1.1 of this document. This document advises on best practices to support future planned developments in DDI design rules. Additional items not added to the Best Practices Document will be placed in High-Level Documentation and/or be used in defining the product level support for various applications.

DDI 3.3 entry and review

DDILIFE issues which were filed as part of the DDI Lifecycle 3.3 review were processed.

Of the 53 issues identified during the review process:

No.	Status	Description
3	Open	creation of the change log, determination of package content and readme.txt update
3	In Process	creation of a validation test (not required for 3.3 publication); documentation of methodology section – post meeting
3	Resolved	Resolved during meeting requiring entry
3	Resolved	Corrections in examples requiring finalization of 3.3 content to complete – post meeting
19	Closed	Closed prior to October meeting
32	Closed	Entered into the 3.3 schema, reviewed and closed during meeting

TC Issues Update

At the beginning of the meeting there were 19 TC issues unresolved. These were reviewed and discussed as follows:

Issue No.	Topic	Status Action [notes]
TC-201	CV Production: decisions and implementation regarding versioning, production and usage	No change [See notes above for progress during meeting]
TC-157	This is what the issue on primitives is about...the appropriate UML primitives and the appropriate translation to bindings. I'll link the issues and transfer the information to make sure it is not lost.	No change
TC-4	ARPA registration of DDI URN	No change
TC-1	Define process for migration from Drupal to COGS - this has been pretty well determined by the EA-> Canonical XMI -> csv file -> COGS however this only addresses what was put into the Core - do we lose the rest or do we capture it in COGS in some way	Changed to In Progress [discussed where issues were in terms of content and approaches – not critical to DDI4 production for review]
TC-208	Generic agency value - add information to Best Practices document	Change to Closed
TC-207	Review use of common properties across versions	Changed to In Progress [generate from finalized 3.3, 2.5 and DDI4 to identify during mapping and integration process]
TC-205	Overall alignment across DDI versions	Changed to In Progress [Flavio is exploring higher level conceptual model of DDI to aid in this work as part of integration and mapping]
TC-204	Alignment of classification models across DDI versions and XKOS	Changed to In Progress [Priority item in mapping]
TC-203	High level feedback DDI4 (NDS) - level of interoperability between standards (line of DDI	Added cross-version label [interoperability work]

	development); how do the lines hold together?; how does DDI4 fold into continued development	
TC-189	Feedback from ICPSR staff committee on DDI4 prototype - interoperability across models; migration is a major issue; possible use of underlying conceptual model	No change [Need to tie into integration draft]
TC-167	Specification of default layout properties at the file level - how should these be recorded and relayed to developers/implementers?	Added labels [This will be part of developers information for DDILifecycle 3.3]
TC-78	Ensure that source target cardinalities are flipped when content is transferred to COGS	No change [per Oliver to ensure this is correct in both input and output]
TC-3	Recommended validity check for DDI4 - Valid realization of pattern (in relation to COGS requirements)	Changed to In Progress
TC-209	COGS interpretation of content and resulting XMI	Changed to In Progress [role of COGS created XMI see discussion under COGS]
TC-163	Document information - primarily the idea of serialized information, expressions that are transitory, expressions used for preservation, transfer, etc.	No change
TC-80	Incorporation of UML class relationship information to COGS	Changed to In Progress [role of COGS created XMI see discussion under COGS]
TC-55	Error: PSM for XSD has identical xmi:id - is this still a valid issue	Changed to Closed prior to meeting
TC-210	Face-to-Face meeting	No change [will close when draft documents from meeting are completed]

Scientific Board Direction and Goals

- What are the goals?
- What is the work plan?
- Future of the Moving Forward project?
- Discussion

What are the goals?

- **Consolidation** in the main ongoing activities
  - Scientific Board Restructuring
  - DDI Lifecycle
  - DDI Cross Domain Integration
  - Training
- Best practices on …
  - When to use which specification, and which part?
  - Portability of DDI metadata between specifications and to outside specifications
  - Could relate to the CMM - CESSDA Metadata Model

Other Goals

- Publication of DDI-RDF Discovery Vocabulary (Disco)
- Working group on SDTL
  - Structured Data Transformation Language (SDTL) is an independent intermediate language for representing data transformation commands. Statistical analysis packages (e.g., SPSS, Stata, SAS, and R) provide similar functionality, but each one has its own proprietary language.

Plans: Scientific Board Restructuring

- Finalization of proposal
- Discussion and approval
- Change of bylaws
- Election of new Scientific Board
- Founding meeting

Plans: DDI Lifecycle

- See presentation of TC

Plans: DDI Cross Domain Integration

- Online meetings with domain-specific groups and domain-agnostic standards. Purposes are information and involvement in review.
- Improvement based on public review
- Intensive review in Dagstuhl workshop in Oct 2020 (?)
- Publication in early 2021
- Face-to-face meeting in 2021 on application of DDI-CDI as integration tool in interaction with other metadata standards
- Cooperation with CODATA, Committee on Data of the International Science Council (ISC)
  - Dagstuhl workshops on „Interoperability of Metadata Standards in Cross-Domain Science, Health, and Social Science Application “, Oct 2020 (?), 2019 report, 2018 report
  - Working groups in relationship to CODATA decadal program
  - *Topic belongs to the strategic goal „Engagement with Global Digital Research Infrastructure”*

Status of Moving Forward Project

- Created for doing research and development regarding the next generation DDI 4
- The realization turned out to be ambitious. The original goals changed.
- „DDI 4 Modeling Group“ developed to „MRT - Modeling, Representation, and Testing Lifecycle Working Group”
- Visible results of Moving Forward efforts
  - DDI Cross Domain Integration
  - Some findings are finding their way into DDI Lifecycle
- Task of MRT developed to two activities:
  - Improvement of DDI-CDI
  - Organization of distribution / cooperation regarding DDI-CDI

  MRT might become a more generic CDI working group

Proposal:
Training as Comprehensive Bracket

- Training could be seen as **glue between all activities**
  - DDI-L, DDI-CDI, Controlled vocabularies, marketing, and website
- Training library with specification-agnostic versus specification-specific material
- Reusable parts for training and documentation of specifications: mutual improvement and benefit
- Focus on online materials
  - Self-teaching slides with explaining text
  - Videos
- Training could be understood as marketing in a wider sense

Further resources on training: wiki, report at annual meeting 2020, Executive Board 2020-04-01

Corona Impact

- Face-to-face meetings are not possible for an unforeseen time
  - Relates to planned meetings of TC, MRT, tutorials at conferences
- Idea: Rededication of funding for planned travel /meeting
  - Could be used for paid support to improve training material and documentation in a comprehensive sense

What is the work plan?

- Several tasks are already mentioned
- Work plan needs discussion especially on …
  - Training as comprehensive bracket (especially in Corona times)
  - Best practices on specification use and portability of metadata

Resources of the DDI Alliance

- Financial resources coming from membership fees, resulting in approx. 100,000 USD per year
  - Major increase of membership is not realistic
- Volunteering work
  - In-kind contributions of representatives of member organizations
  - Contributions of interested experts
- Other in-kind contributions
  - Organization of training workshops
- Cooperations with other organizations?

The work in working groups and sprints rely mostly on **volunteering work**. This is the most **valuable resource**.

Alliance culture – an asset to be protected. (external review 2011)

*„The culture is characterised by an **extraordinary level of trust and collaboration** that was clearly evident throughout this review. This is one of the Alliance's greatest assets going forward and has had no small part in its success to date. The candour and openness whether in group or one-on-one discussion is, in our experience, rarely observed and even more rarely consistently realized. The core values of the Alliance don't need fixing and **must be actively fostered and preserved going forward**."*

DDI Alliance Controlled Vocabularies Working Group

Activity Report

2019-2020

Current members:

Sharon Bolton, United Kingdom Data Service (UKDS)

Taina Jääskeläinen, Finnish Social Science Data Archive (FSD)

Alexander Jedinger, GESIS - Leibniz Institute for the Social Sciences

Hilde Orten, Norwegian Centre for Research Data (NSD)

Sanda Ionescu, ICPSR, University of Michigan, U.S. – lead

Lisa Isaksson, Swedish National Data Service (SND)

Accomplishments:

The CESSDA Vocabulary Service was successfully launched in 2019. This is an online interactive tool for creating, editing and publishing controlled vocabularies. It supports versioning and translations of the vocabularies as well as separate versioning of the translated lists. The tool includes a public interface for searching and browsing all of the published vocabularies as well as their translations in various European languages, where available. The vocabularies may be downloaded in several different formats – PDF, SKOS and HTML.

DDI-CVG will be using this service in our work moving forward. To this end, in the past year we have re-published all of the DDI Alliance vocabularies in the CESSDA tool and, in the process, we have also reviewed each of the CVs for potential problems that required resolving before publication.

As a result, we have published new and improved versions of six CV lists: Aggregation Method, Analysis Unit, Data Type, Numeric Type, Type of Address, and Type of Note.

We have also published a new controlled vocabulary, for Contributor Role.

Goals for next year:

We are now in the process of preparing a new version of the CV for Mode of Collection.

We will be updating the Controlled Vocabularies pages on the DDI Alliance site to make them consistent with the changes operated within the Vocabulary Service tool, including the new formats available for download.

We will continue to create new vocabularies and update the existing ones as needed.