DPM 2.0 – Refit project
Outline

1. DPM background (EBA-EIOPA collaboration)
2. EBA experience with DPM and DPM Refit role towards integration
3. EIOPA experience with DPM and DPM Refit uses
4. EC supervisory data strategy and links with DPM
6. DPM 2.0 timelines
The EBA Data Point Model development started in 2012, to support the EBA reporting framework 2.0.

The DPM database was first published by end 2013, as part of the ITS 2.0 technical package.

Over the last 8 years, the DPM database has been accumulating all the successive releases of the EBA data dictionary, from version 2.0 to 3.3.

The data dictionary tracks changes and maintains the full historization of all templates structure, data points categorisation, validation rules, and taxonomies, across all releases.

The core structure of the DPM database has not changed significantly since its first publication.

The DPM database is being used as a main component of EUCLID (as it had been already with the previous ESP reporting system).

The DPM database is also at the core of the EBA solutions for regulatory data analysis.

The EIOPA on 2011 decided to implement the xBRL format for data exchange of regulatory reporting data.

The Data Point Model development started in 2012, to support the business & technical development of the Solvency 2 reporting framework.

The first DPM model was published on 2013, both on Excel and xBRL formats.

In 2014 EIOPA developed the Tool for Undertakings to support the insurance companies on the creation of Solvency 2 on xBRL, adopting the EBA’s DPM database as core central piece of the software solution.

In 2015 EIOPA published the Solvency 2, DPM and xBRL taxonomies.

The EIOPA DPM has being evolving since then, covering in the single glossary all the EIOPA’s reporting frameworks, including the ECB add-ons (insurance and pension funds) with the define-once approach.
DPM background

EBA-EIOPA collaboration

Use of DPM methodology

Integration through a single data dictionary

Building a metadata driven strategy

Facing challenges together

Working together in the development

Building a common long-term goal

The EBA and EIOPA collaboration on data standardisation since both authorities started to use DPM methodology.

Both ESAs have being very successful on integrating their regulatory frameworks using a single data dictionary.

Both, adopted the metadata-driven strategy as the way to respond to changes and reduce costs.

Similar future challenges, made evident the advantages of more strong collaboration and harmonisation.

Experts from both ESAs have been working on the new common data model, envisaging the common development of Data Definition standards and tools.

As a long-term goal, this should play a key role to enable a semantic integration of a single financial dictionary for the whole financial sector.

Total convergence of EBA and EIOPA methods, models, processes, and tools used for the development of data dictionaries and related regulatory products.
DPM Refit project goals

**Total convergence of EBA and EIOPA** methods, models, processes, and tools used for the development of data dictionaries and related regulatory products

**Unified and versatile metamodel** applicable to all regulatory data exchanges, from highly aggregated data points to very granular data sets of prudential, statistical or transactional information

**Content extensible and interoperable** for defining, reusing and exchanging metadata for regulatory data requirements

Enabling the possibility of subsequent **semantic integration** of data dictionaries across different regulatory domains
The EBA experience was guided by:

- Use the DPM, as a unique common data dictionary including all elements needed to define the regulatory data
- Publish always the regulatory text accompanied with formal and standard data definitions
- Produce a standardised technical package fully aligned with regulation and envisaging its direct use by CAs and institutions in their digital processing
- Enable systems communication by sharing DPM as a common platform of understanding across different areas and users

The goal of integrating the EBA reporting frameworks was always a priority in the definition of supervisory and resolution regulation.

The DPM experience

DPM Refit role towards further integration

- Standard and comparable definitions
- Ready to support digital processing
- Ready to support systems connectivity

- Review of the data model and glossary
- Integrated reporting and works towards semantic integration – work with the ESAs, ESCB, SRB and EC
- Pilar III data hub and pillar 3 integration with reporting
The DPM implements the uniform and consistent definitions included in the implementing technical standards (ITS), guidelines and Board of Supervisors decisions on reporting and disclosure.

- Provides a structured representation of the information, identifying all the business concepts and their relations, as well as validation rules.
- One model for all data reporting requirements under EIOPA remit (insurance, pensions, PEPP providers, public disclosure) and for the ECB reporting extensions.
- The model facilitates the appliance of waivers and completeness information reporting via the basic information – Master data.
- The common syntactic format will further support data harmonisation for both EBA and EIOPA, providing synergies for a common development of tools for the digital regulatory reporting.

Technical and business changes affect the outputs expected hence require communication and gradual implementation to allow preparation.

What’s next for EIOPA on DPM Refit:

- Impact on taxonomies and data exchange
- Impact on insurance ECB add-ons
- EC data strategy
- Business changes (Financial Conglomerates, Decision on IORPs, 2020 Review, EU-US agreement)
- Monitoring development of new formats (CSV)
- Use of platform Atome:Matter
- Digital Reporting Tool under development

The EIOPA architecture outline on DPM:
- Solvency II group and solo reporting
- Financial stability
- Pillar III public disclosure templates
- IORP reporting
- PEPP reporting
- ECB insurance and pension funds add-ons

EIOPA Evolution of regulatory reporting with the current data point model
EC supervisory data strategy and links with DPM

**EU Supervisory Data Strategy**

- Facilitate digital transformation
- Consistent and standardised data
- Improved design of reporting requirements
- Data sharing and reuse

- Data driven policy and decisions
- Greater standardisation across sectors
- Facilitates multiple use and re-use
- Contributes to metadata-driven processes
- Powered by more accessible and efficient technology
- Integrated and embedded into internal data
- More efficient and less costly to produce and deliver

Facilitate digital transformation
- Consistent and standardised data
- Improved design of reporting requirements
- Data sharing and reuse
Most of the building blocks are done/decided. From now only minor changes can be expected mainly due to experience gained when doing the migration of metadata and software.

Two building blocks still ahead to be developed: interoperability and governance.

EBA, EIOPA and other EU component authorities are analyzing the following DPM Refit implementation phases.
**Interoperability API**

- **Goal:** to enable meta-exchange among DPM Refit repositories
- **Decentralized approach**
- **Based on open standards (WS/REST)**
- **First version read only**
- **Based on DPM Refit ID and GUIDs to enable cross-domain interoperability**

**Governance**

- **Clear and transparent decision making process for the maintenance and evolution of the DPM Refit**
- **Open standard to European and National competent authorities and other standard user stakeholders**
- **Focused on the scope of DPM methodology (syntactical integration)**
- **Focused on operational aspects and complementary to the promotion and maintenance of the Refit as ISO standard**

**Process has not started yet, so all still to be discussed**
## What is next on EIOPA’s DPM implementation

<table>
<thead>
<tr>
<th>Taxonomy - application date</th>
<th>Publication</th>
<th>Business changes</th>
<th>Current approach</th>
<th>Atome</th>
<th>DPM Refit</th>
<th>CSV</th>
<th>DPM Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8.0 – 12.2023</td>
<td>Current approach: July 2022 Validations added in Jan 2023 only via ATOME: Matter.</td>
<td>SII (new ITS on reporting and disclosure)</td>
<td>Yes – for the dictionary and annotated templates for comparison No – for validations would be only provided in ATOME: Matter outputs</td>
<td>Yes, for information and adaptation; Validations only from ATOME: Matter</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2.8.1 - 12.2023?</td>
<td>PWD in July 2023 (tentative)</td>
<td>Financial Conglomerates (FICOD)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2.9.0 – 12.2024 (or Q12025?)</td>
<td>PWD1 in Q1 2023 (tentative)</td>
<td>IORPs Decision</td>
<td>No</td>
<td>Yes</td>
<td>Yes, for comparison purposes only</td>
<td>To be confirmed (only if adequate engines are available to execute rules/validations in csv)</td>
<td>No</td>
</tr>
<tr>
<td>2.10.0 – 12.2025</td>
<td>1 June 2024 without validations; 15 July 2024 with validations</td>
<td>SII Review</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No - Potentially for info</td>
</tr>
<tr>
<td>2.11.0 - 12.2026</td>
<td>1 June 2025 without validations; 15 July 2025 with validations</td>
<td>If needed</td>
<td>No</td>
<td>Yes, for comparison</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Following years</td>
<td>If needed</td>
<td>No</td>
<td>No (to be confirmed)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
**Main Milestones**

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Target date</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM Refit finalised</td>
<td>End 2022</td>
</tr>
<tr>
<td>DPM Studio delivered</td>
<td>Q2 2023</td>
</tr>
<tr>
<td>Calculation/Validation engine delivered</td>
<td>Q2 2023</td>
</tr>
<tr>
<td>DPM semantic review</td>
<td>Q3 2023</td>
</tr>
<tr>
<td>DPM content migration from old to new model</td>
<td>Q3 2023</td>
</tr>
<tr>
<td>Start using DPM Studio/DPM Refit for the development of new Framework releases</td>
<td>Q3 2023</td>
</tr>
<tr>
<td>Internal dependent EBA systems adapted to DPM Refit</td>
<td>Q4 2023</td>
</tr>
<tr>
<td>Start DPM transition period, where new DPM releases are published both in the old and new formats</td>
<td>Start 2024</td>
</tr>
<tr>
<td>End DPM transition period, aligned with XBRL transition period. Only the new DPM refit will be published, and only XBRL-CSV files will be accepted by the EBA</td>
<td>End 2025</td>
</tr>
</tbody>
</table>
The DPM in the EBA reporting ecosystem

DPM studio
Data definition
Rules definition

DPM refit
Data Dictionary
Validations/Calculations

EDAP
Data Dissemination

EUCLID MDM
Master Data Mgmt
Calendar Mgmt

DWH
Data storage
Data preparation

EUCLID RRP
Regulatory Data Collection

SSBI
Self-Service Analytics

CVE
Calculation and Validation Engine

SAS
Statistics and Analytics
DPM 2.0 Refit project – Metamodel Intro
Outline:

1. DPM meta-model overview
2. Meta-model metadata
3. Administration and documentation
4. Historization: releases, deactivations, application dates
5. Glossary: basic and advanced modelling concepts
6. Representation of information requirements
   - Application of glossary terms
   - Rendering: tables, headers, cells
   - Role and definition of variables
   - Packaging: frameworks, modules, table groups
7. Operations: data quality checks and data derivation rules
Metamodel: Metadata

### Class
- **Chapter**
  For a publication that uses chapters, this part should be used to capture this information. Because chapters are not necessarily numbers, this is a string.

### Organisation
- **Article**
  Article refers to a statutory article in legal material.

### Property
- **Subsection**
  Subsection refers to a subsection of the section part.

### TextExcerpt
- **Subdivision**
  Subdivision is used to refer to specific subdivisions in a document.

### Operator
- **Order**
  Order is used to order items in a list.

### Operation
- **Function**
  Function is used to perform a specific operation on data.

### Attribute
- **DPMClass**
  DPMClass is used to represent a class in a DPM model.

### PK ClassID
- **Name**
  Name is used to identify a class.

### FK OwnerClassID
- **HasReferences**
  HasReferences is used to indicate a relationship between classes.

### PK LanguageCode
- **Language**
  Language is used to specify the language of a document.

### PK SubdivisionTypeID
- **SubdivisionType**
  SubdivisionType is used to indicate a subdivision in a document.

### PK OperatorID
- **Operator**
  Operator is used to define an operator in a logical expression.

### PK ArgументID
- **OperatorArgument**
  OperatorArgument is used to define arguments for an operator.

### PK SubcategoryItemID
- **SubcategoryItem**
  SubcategoryItem is used to define a subcategory item in a classification system.

### FK ClassID
- **FK ClassID**
  FK ClassID is used to reference a class in another class.

### FK OwnerClassID
- **FK OwnerClassID**
  FK OwnerClassID is used to reference the owner class in a relationship.

### PK AttributeID
- **PK AttributeID**
  PK AttributeID is used to reference an attribute in another class.

### HasReferences
- **HasReferences**
  HasReferences is used to indicate a relationship between classes.

### PK DataTypelD
- **PK DataTypelD**
  PK DataTypelD is used to reference a data type in another class.

### FK ParentDataTypelD
- **FK ParentDataTypelD**
  FK ParentDataTypelD is used to reference the parent data type in a relationship.

### Code
- **Code**
  Code is used to represent a code in a classification system.

### Subcategory
- **Subcategory**
  Subcategory is used to define a subcategory in a classification system.

### Category
- **Category**
  Category is used to define a category in a classification system.

### SubcategoryItem
- **SubcategoryItem**
  SubcategoryItem is used to define an item in a subcategory.

### PK SubdivisionTypeID
- **PK SubdivisionTypeID**
  PK SubdivisionTypeID is used to reference a subdivision type in another class.

### FK SubdivisionTypeID
- **FK SubdivisionTypeID**
  FK SubdivisionTypeID is used to reference a subdivision type in another class.

### Language
- **Language**
  Language is used to specify the language of a document.

### PK LanguageCode
- **PK LanguageCode**
  PK LanguageCode is used to reference a language code in another class.

### SubdivisionType
- **SubdivisionType**
  SubdivisionType is used to define a subdivision type in a classification system.

### PK SubdivisionTypeID
- **PK SubdivisionTypeID**
  PK SubdivisionTypeID is used to reference a subdivision type in another class.

### PK OperatorID
- **PK OperatorID**
  PK OperatorID is used to reference an operator in another class.

### PK ArgumentID
- **PK ArgumentID**
  PK ArgumentID is used to reference an argument in another class.

### PK DataTypelD
- **PK DataTypelD**
  PK DataTypelD is used to reference a data type in another class.

### FK ParentDataTypelD
- **FK ParentDataTypelD**
  FK ParentDataTypelD is used to reference the parent data type in a relationship.

### DataTypelD
- **DataTypelD**
  DataTypelD is used to define a data type in a classification system.

### PK DataTypelD
- **PK DataTypelD**
  PK DataTypelD is used to reference a data type in another class.

### FK ParentDataTypelD
- **FK ParentDataTypelD**
  FK ParentDataTypelD is used to reference the parent data type in a relationship.

### Operator
- **Operator**
  Operator is used to define an operator in a logical expression.

### PK OperatorID
- **PK OperatorID**
  PK OperatorID is used to reference an operator in another class.

### PK ArgumentID
- **PK ArgumentID**
  PK ArgumentID is used to reference an argument in another class.

### PK DataTypelD
- **PK DataTypelD**
  PK DataTypelD is used to reference a data type in another class.

### FK ParentDataTypelD
- **FK ParentDataTypelD**
  FK ParentDataTypelD is used to reference the parent data type in a relationship.
**Concepts: Owners**

*Predefined organisations:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Acronym</th>
<th>IDPrefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM Metamodel</td>
<td>DPMM</td>
<td>100</td>
</tr>
<tr>
<td>European Banking Authority</td>
<td>EBA</td>
<td>101</td>
</tr>
<tr>
<td>European Insurance and Occupational Pensions Authority</td>
<td>EIOPA</td>
<td>102</td>
</tr>
</tbody>
</table>

*Implementation aspect: IDPrefix to ensure global uniqueness of IDs (to facilitate physical merging to models in one database)*
Enables translating various attributes (Names, Descriptions, Labels, Values, Texts, ...) to various languages by various organisations (there can be multiple translations in one language by different organisations)

Enables representation of Expressions of Operations in various syntaxes/formats
Ability to identify references of created concepts within any document structure
Handling legal references (e.g. to regulations, standards) but also change requests

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter</td>
<td>For a publication that uses chapters, this part should be used to capture this information. Because chapters are not necessarily numbers, this is a string.</td>
</tr>
<tr>
<td>Article</td>
<td>Article refers to a statutory article in legal material.</td>
</tr>
<tr>
<td>Section</td>
<td>Section is used to capture information typically captured in sections of legislation or reference documents.</td>
</tr>
<tr>
<td>Subsection</td>
<td>Subsection is a subsection of the section part.</td>
</tr>
<tr>
<td>Paragraph</td>
<td>Paragraph is used to refer to specific paragraphs in a document.</td>
</tr>
<tr>
<td>Subparagraph</td>
<td>Subparagraph of a paragraph.</td>
</tr>
<tr>
<td>Clause</td>
<td>Subcomponent of a sub paragraph.</td>
</tr>
<tr>
<td>Subclause</td>
<td>Subcomponent of a clause in a paragraph.</td>
</tr>
<tr>
<td>Appendix</td>
<td>Refers to the name of an Appendix, which could be a number or text.</td>
</tr>
<tr>
<td>Example</td>
<td>Example captures examples used in reference documentation; there is a separate element for Exhibits.</td>
</tr>
<tr>
<td>Page</td>
<td>Page number of the reference material.</td>
</tr>
<tr>
<td>Exhibit</td>
<td>Exhibit refers to exhibits in reference documentation; examples have a separate element.</td>
</tr>
<tr>
<td>Footnote</td>
<td>Footnote is used to reference footnotes that appear in reference information.</td>
</tr>
<tr>
<td>Sentence</td>
<td>In some reference material individual sentences can be referred to, and this allows them to be referenced.</td>
</tr>
<tr>
<td>URI</td>
<td>Full URI of the reference such as “<a href="http://www.fasb.org/fas133%E2%80%9D">http://www.fasb.org/fas133”</a>.</td>
</tr>
<tr>
<td>Requirement</td>
<td>A suggestion of a new model entry for consideration / to be addressed in the next releases.</td>
</tr>
</tbody>
</table>
Relating concepts: two or many

Relation types can be directed (IsRelated pointing to relation target) or undirected (working both ways)

Predefined relation types for:

- **Variables** - “factVariable_keyVariable” and “variable_attributeVariable”
- **SubCategories** - “subCategoryMaster_version” and “subCategoryRendering_version”
- **Tables** - “table_variant”

Generic relation types:

- “equivalent_concept”
- “version_fix”/“version_new”
**Historization: Releases**

**Release** – publication of the model

Indicated on objects or connections between objects (to enable changing composition)

*Start Release* is mandatory, *End Release* is optional

does not address model administration purposes (e.g. creation/modification dates), workflow/development process stages (IWD, PWD, ...) neither it helps to handle indication of temporary metadata (work-in-progress for internal comments/feedback)
**Historization: Deactivation**

**Categories** and **Items** (therefore also **Properties** for this a counterparty **Item** is created) can be marked as deactivated (e.g. no longer to be used)

Metamodel authors can decide on deactivating certain **DataTypes**
**ModuleVersions** may have their own life cycle with application dates at some point in the future (e.g. module applicable starting from next year).

**OperationVersions** can be activated or deactivated (or have their severity modified) starting from specified submission date (i.e. applicable to all reports exchanged after certain day).
Change Log (non-normative)

non-normative: not aimed to become part of the DPM standard

stores modifications made to the content of model entities or their attributes (when, what, who, why)

implementation aspect: `RowGUID` on every model entity to enable single reference
Mapping to current DPM:

- Domain ~ Category
- Metric ~ Property
- Member ~ Item
- Dimension ~ Property
- Hierarchy ~ SubCategory

New:

- Super Category
- Compound Item

- Historisation (registering changes in composition of Categories, versioning of SubCategories, ...)
Glossary: Categories, Properties and Items

Categories help organising Glossary:

- **Enumerated Categories** group **Items** that share common nature/semantics
  - Item can change Category

- **Not enumerated Categories** gather data-type-constrained **Properties**
  whose values are impractical or impossible to enumerate (e.g. volatile)

Properties:

- **quantitative metrics** *(IsMetric)* - identify what is measured, determine data type and time duration of observation

- **qualitative** - provide perspective/characteristic; enumerated properties contextualise **Items**

- **Property** can change Category

In physical implementation: each **Property** has its counterpart **Item** (enables arranging **Properties** in **SubCategories** to be used as dropdowns)
### Glossary:

#### Predefined Categories

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Category</th>
<th>Enumerated</th>
<th>Items</th>
<th>SubCategory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location of activity,</td>
<td>enumeration</td>
<td>Countries</td>
<td>yes</td>
<td>Poland, Spain, Greece, France, China…</td>
<td></td>
</tr>
<tr>
<td>Counterparty residence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISIN, CUSIP, SEDOL, …</td>
<td>string</td>
<td>Instrument codes</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Location of activity, Counterparty residence, ISIN, CUSIP, SEDOL, Source of information, Name of subsidiary, Carrying amount, Type of instrument code, …</td>
<td>1: ISIN/CUSIP/SEDOL/…</td>
</tr>
<tr>
<td>Source of information,</td>
<td>various</td>
<td>Not applicable</td>
<td>no (no meaning)</td>
<td>From Google, From Facebook, From in person events, From a friend, From newspapers, …</td>
<td>1: Google/Facebook/In person event/…</td>
</tr>
<tr>
<td>Type of instrument code, …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carrying amount, Name</td>
<td>Amount types, Not applicable, none</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of subsidiary, …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Open and granular data challenges:

<table>
<thead>
<tr>
<th>Code Type</th>
<th>Code Value</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIN</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>CUSIP</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>SEDOL</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Amount</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code Type</th>
<th>Code Value</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIN</td>
<td>CUSIP</td>
<td>SEDOL</td>
</tr>
</tbody>
</table>

### Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PR</td>
<td>Properties</td>
<td>Contains Items which are counterparts of Properties in physical implementation of the DPM metamodel.</td>
</tr>
<tr>
<td>_NA</td>
<td>Not applicable</td>
<td>Contains Items which do not belong to any specified Category such as those that are typically used only in dropdowns on Headers or Variables. It is also linked by Properties that do not belong to any real Category. Such enumerated Properties can refer to Items from Categories: &quot;Not applicable&quot;, &quot;Properties&quot; and in such case they can also use Items of other Categories, in particular by being linked to such mixed SubCategory. It is not a semantically meaningful Category therefore it is neither enumerated nor not-enumerated.</td>
</tr>
<tr>
<td>_TE</td>
<td>Templates</td>
<td>Contains Items which represent Templates (TableGroups or Tables) for purposes of resembling Filing indicator Variables.</td>
</tr>
</tbody>
</table>
SubCategories are lists of Items but enable also their arrangement in hierarchies (if needed, e.g. to reflect nesting or arithmetic relationships – by reference to Operator)

- document and put Items in order
- can be used as dropdowns by Table Headers (and hence also Variables)

SubCategories can change composition (through SubCategoryVersion)

SubCategories may be related to one another (via ConceptRelation)

- e.g. a “list of all countries” can be indicated as a “master”/“full version” for the regional/topic oriented subsets (which may need to be updated once the “master” list is updated)

- For rendering purposes (ability to show a complete breakdown from which only certain options are selectable);
SuperCategory is a union of Categories

- e.g. frequently requested mix of breakdowns by “Countries” and “International organizations” (i.e. IMF, BIS, ECB, ... along countries for exposures/risk classification),
- can combine enumerated and non-enumerated Categories

reuses Items of other Categories (or even SuperCategories) and can introduce its own Items (e.g. for “total”)

may have its own Properties

composition of SuperCategory is versioned (can change between Releases)
Glossary: Compound Items

Compound Items

- are composed of Property-Item pairs – simplified representation of complex terms

<table>
<thead>
<tr>
<th>Instrument type</th>
<th>Treasury bills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instrument type</td>
<td>Debt securities</td>
</tr>
<tr>
<td>Issuer sector</td>
<td>General Government</td>
</tr>
<tr>
<td>Original maturity</td>
<td>&lt; 1 year</td>
</tr>
</tbody>
</table>

- they may belong to existing Categories (e.g. of one of the contributing Items)

- their composition is versioned

implementation reuses Context and its Composition
Properties are used on

- **TableVersion** (applicable to entire table)
- **HeaderVersion**
- **Variable** (mandatorily)

For dropdowns the above are associated to **SubCategoryVersion**

Optionally, **Property-Item** pairs gathered in **Contexts** providing additional metadata can be associated to any of the above
Representation of information requirements

Rendering: Tables, Headers, Cells, Association and KeyHeaders mapping

Grouping: TableGroups

Identification of information requirement: Variables (Key, Fact, Attribute, Filing indicator and their linkage)

Packaging: Frameworks, Modules
Tables have Headers (~Ordinates) and Cells

Tables can be grouped

Tables can be associated to one another
**TableVersion** may link to *Property* and/or *Context* (*Property-Item* pairs) applicable/common to/shared by all *Table Headers* and *Cells*

**Headers Direction** can be *Rows/Columns/Sheets* - one set of related headers per disposition (note: typical open tables require only columns)

**HeaderVersion:**
- contribution to definition of a *Fact Variable*: column/row/sheet
- *Headers* may point to *Property* and/or *Context*
- definition of a *Key Variable*
  - points to *Property* and may point to *SubCategory*
  - does not link/result in a *Cell*
  - links to *VariableVersion*

Implementation aspect: introduction of **TableVersionHeader** enables modification of headers structure without a need for duplicating objects for the entire table (all its *Headers*, *Cells*, etc)
Cell points to at least one leaf-level Header (e.g. a column in open rows table) but may also point to two or maximum three (one for each: Column, Row and Sheet)

on TableVersionCell it can be indicated if a cell is mandatory or excluded (this information may change for a table between table versions/releases)

TableVersionCell definition results from intersection of Headers for this Cell (including inheritance) and points to Fact Variable unless combination of characteristics from this intersection is illogical (e.g. “Equity instruments” issued by “Government” with certain “Maturity period”)
TableGroups may be created for various purposes as indicated by TableGroup.Type:

- “templateGroup”,
- “template”,
- “templateVariant”,
- “templateScope”.

TableGroups of Type “templateScope” can also be nested: DPM XL syntax enables using their Codes in Operation.Expression instead of Table Codes.
Non-normalized Tables do not need to be modelled when they are IsAbstract, in which case they are broken down in two or more modelled Tables (abstract Tables, if modelled, do not contain links to Glossary terms, just pure rendering)

Tables can be Associated to one another, indicating FKS mapping, Cardinality, Subtyping (with Discriminator), etc.
**Information requirements: role of variables**

**Variable** represents each distinct reportable value (regardless of its occurrence in rendering) – quantitative or qualitative

- **types:**
  - **Fact** (monetary amount, text, enumeration, ...) ~ DataPoint
  - **Key** (e.g. column identifying each row in an open table)
  - **Attribute** (unit of measure, value precision/accuracy, a comment of a Fact or a Key)
  - **Filing indicator** (indicator that certain subset of information, typically a Table or a set of Tables is intentionally contained or not contained in a report)

- **ConceptRelations**: **Variable-Attribute, Equivalent-Variable** (same meaning but modelled differently), ...

**VariableVersion**:

- must refer to **Property**
- may refer to **SubCategoryVersion** (list of possible **Items**) or to **Context** (composed of one or many **Property-Item** pairs)
- composition may change between **Releases** (changes in modelling, fixing of bugs, etc)
Fact Variables can be derived from Table Cells: Table level glossary terms contribute to definition of Table Headers (where information can be inherited from upper levels) and is further propagated to Cells on intersection of these Headers.

Key variables are derived from Headers which have no cells attached (this is to ensure open/semi-open sheets and columns are modelled in the same way)

- TableVersion refers to Key that though KeyComposition identifies Variables that are keys in a given table.
- KeyComposition arranges Keys in sets to support rendering-less models.

Filing indicator variables are derived from Cells of Tables using Items of dedicated Category.

ModuleParameters apply globally for Module, for example Attribute Variables such as report currency and period/date.

ModuleVersion

PKModuleVID
FK Module VID
FK GlobalKeyID
FK StartReleaseID
FK EndReleaseID
Code
Name
Description
VersionNumber
FromReferenceDate
ToReferenceDate
FK RowGUID

ModuleParameters

PK ModuleVID
FK Module VID
FK VariableVID
RowGUID

PKVariableID

Type
FK RowGUID

PKVariableVID
FK VariableID
RowGUID

PKVariableID

FK PropertyID
FK SubCategoryVID
FK ContextID
FK KeyID
IsMultiValued
Code
Name
FK StartReleaseID
FK EndReleaseID
FK RowGUID

PKKeyID

FK RowGUID

FK RowGUID

PKContextID

U Signature
FK RowGUID

PKContextID

U Compromise
FK ParentDataTypeID
FK Active
FK RowGUID

PKDataTypeID

U Code
U Name
FK ParentDataTypeID
FK RowGUID

ModuleTypeID

PK Module VID
FK Module VID
FK Variable VID
RowGUID

TableVersion

PK Table VID
FK CellID
CellCode
IsNullTable
IsExcluded
IsVoid
Sign
FK Variable VID
RowGUID

TableVersionCell

PK Table VID
FK CellID
CellCode
IsNullable
IsExcluded
IsVoid
Sign
FK Variable VID
RowGUID

TableVersion

PK Table VID
Code
Name
Description
FK TableID
FK AbstractTableID
FK KeyID
FK PropertyID
FK ContextID
FK StartReleaseID
FK EndReleaseID
FK RowGUID

PKContextID

FK ContextID
FK StartReleaseID
FK EndReleaseID
FK RowGUID

PKContextID

FK PropertyID
FK SubCategory VID
FK ContextID
FK Variable VID
FK StartReleaseID
FK EndReleaseID
FK RowGUID

PKVariableID

FK PropertyID
FK SubCategory VID
FK ContextID
FK Variable VID
FK StartReleaseID
FK EndReleaseID
FK RowGUID

PKVariableID

FK PropertyID
FK SubCategory VID
FK ContextID
FK Variable VID
FK StartReleaseID
FK EndReleaseID
FK RowGUID

KeyComposition

FK KeyID
FK Variable VID
RowGUID

FK RowGUID

FK RowGUID

FK RowGUID

FK RowGUID

FK RowGUID

FK RowGUID

FK RowGUID

FK RowGUID
**Framework** – regulatory reporting requirements organized by subject/thematical area

**Module** – set of information requirements to be reported together (in one report)

**ModuleVersion**
- *FromDate* and *ToDate* identify its application dates (may differ from model publication dates i.e. releases)
- Gathers *Tables* (their *Versions*) that are required to be reported together

**ModuleParameters** are *Variables* (e.g. *Attribute* or *Key*) applied globally to all *Facts* (note: in case these global characteristics are *Property-Item* pairs, *Item* is the only option on *SubCategory*)
reflected in metamodel using tree structure (AST) comprising of nodes - operators and operands, the latter referring to terms from glossary, rendering, variables (incl. external sources)
syntax agnostic: able to result from and in different syntaxes
automatically populated (parsed from any/some syntax or defined and reviewed in graphical interface by users) and automatically consumed (to produce rules in e.g. XBRL, SQL, VTL, ...)
**Operations**: enable definition of data quality checks as well as data derivation rules (transformations and computation of Variables)

Operations can be sequenced/nested, e.g. serve as precondition, determine dynamic severity, etc.

**Expression** can be translated to natural language, programming language or various syntaxes (e.g. DPM XL)

**OperationNode**: one of defined Operators or an **OperandReference** to any artefact of a model, e.g. Variable, Cell, Item, Property, ...)

- Note: Variables may indicate external data sources (e.g. master/reference data)

**OperationScope** and **Composition** provides flexibility in assignment of Operations to Modules
DPM 2.0 Refit project – Validations & Calculation rules - Intro
DPM validations: Current process

1. User input (semi formal language)
2. DPM DB
3. XBRL formula (in taxonomy)
4. Excel report
5. XBRL engine
6. XBRL XML
EBA and EIOPA have been using a validation language over the years, but without a formal basis and a proper documentation.

The lack of formality causes some challenges:

- It is not possible to have full automation when translating to other languages (notably, to XBRL Formula).
- Although EBA and EIOPA are basically using the same language, there are some differences that further difficult automation and common understanding.
- There may be ambiguity in the meaning of a validation.

The language is translated to XBRL formula, which also adds challenges:

- Not all validations can be translated to XBRL.
- XBRL is very difficult to understand.
- Performance becomes an issue with big instances.
The DPM Refit aims to formalize the operations (including validations and other calculations) by having:

- A formal expression language: **DPM-XL** is a formal language for expressing calculations based on the DPM.
  - **Is based on** the semi-formal language that the EBA and EIOPA have been using to write and share validation rules for several years.
  - In practice, it is the result of a reverse-engineering process to formalize the language that was already existing, with the minimum changes necessary.
- A metamodel to represent the tree of operations, as well as the relations of the operands with the core DPM (**DPM-ML**).

**Why DPM-XL and DPM-ML?** Because:

- DPM-ML is based on variables (stable) instead of rendering (unstable).
- In principle it is possible to translate languages different from DPM-XL into DPM-ML.
The formalisation of the language has three **blocks**:

- **Information model**: Specifies the artefacts that the language is using.
- **Grammar**: Technical definition of the syntax of the language. Allows developers to build parsers for the language (only relevant for DPM-XL).
- **Semantic specification**: Semantics for all operators of the language, specifying formally their constraints and behaviour. For instance, is it possible to use the + operator for two strings?
What are the options the DPM Refit Operations enable?

User input in DPM-XL

Excel report

DPM-ML

DPM-ML

Integrated data

DPM-ML engine

DPM-ML engine

XBRL CSV

XBRL engine

XBRL engine

XBRL CSV

XBRL XML
Outline

Introduction
An example
Information Model
Main operators
Changes to current operators
DPM-ML metamodel
Calculations
### Validation rule

**C 90.00 - Trading book and market risk thresholds (TBT)**

<table>
<thead>
<tr>
<th>Rows</th>
<th>Columns</th>
<th>Month 3</th>
<th>Month 2</th>
<th>Month 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On - and off - balance sheet business subject to market risk</td>
<td>0010</td>
<td>0020</td>
<td>0030</td>
</tr>
<tr>
<td></td>
<td>In % of total assets</td>
<td>51</td>
<td>42</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Total assets</td>
<td>1020</td>
<td>1010</td>
<td>1000</td>
</tr>
</tbody>
</table>

with \{tC\_90.00, r*\}: \{c0070\} * \{c0080\} = \{c0010\}
Any DPM-XL expression can be represented as a tree:

\[
\text{with } \{\text{tc}_90.00, \text{r}^*\} \text{: } \{c0070\} \times \{c0080\} = \{c0010\}
\]
The tree of the expression can be then represented in the DB, with reference to actual DPM variables.
Input data

\{tC\_90.00, r\*, c0010\} →

<table>
<thead>
<tr>
<th>Index</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
</tr>
</tbody>
</table>

\{tC\_90.00, r\*, c0070\} →

<table>
<thead>
<tr>
<th>Index</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5%</td>
</tr>
<tr>
<td>2</td>
<td>4%</td>
</tr>
<tr>
<td>3</td>
<td>6%</td>
</tr>
</tbody>
</table>

\{tC\_90.00, r\*, c0080\} →

<table>
<thead>
<tr>
<th>Index</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1020</td>
</tr>
<tr>
<td>2</td>
<td>1010</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
</tr>
</tbody>
</table>
with \{tC\_90.00, r^*\}: \{c0070\} \times \{c0080\}

with \{tC\_90.00, r^*\}: \{c0070\} \times \{c0080\} = \{c0010\}

<table>
<thead>
<tr>
<th>Index</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>40.4</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
</tr>
</tbody>
</table>
The DPM Expression Language serves to write calculations.

**Calculations** are expressions that use input operands and/or operators to produce a result. Expressions are finite combinations of symbols that are well-formed according to the syntactical rules of the language. Expressions compose some **operands** in a certain order by means of the **operators** of the language, to obtain the desired **result**. The symbols of the expression designate operators, operands, and the order of application of the operators.

Operators specify a type of operation to be performed on some **input operands** (exceptionally, there may be operators that do not take operands as input, e.g., an operator to get the current time) to generate an **output**. The output produced by one operator may be used as input for another operator (i.e., operators can be nested).

**Operands** are specific artifacts from the DPM Expression Language referenced in an expression as input.

The **result** produced by a calculation is also a specific artifact from the DPM Expression Language.
**Scalars** are individual values of a certain *Data Type*.

**Scalar Sets** are sets of Scalar values defined on the same *Data Type*. Scalar Sets are typically used with the *in* operator.
Recordset are collections of Records that share a same Structure. Technically, Recordsets are two-dimensional labelled data structures (tabular), which can be assimilated to Relational Tables or Data Frames. The columns (fields) of the Recordset are provided by the Components of its Structure. The rows of the Recordset are its composing Records.

The Structure of the Recordset is a collection of Components, which can have one of three roles: Key, Fact or Attribute. Each Component has a name, which must be unique within the Recordset.

Each Record of the Recordset is individually identified by the combination of the values for its Key Components.

A Recordset having no Key Components behaves like a Scalar.
Standard Key Components are common to all the Recordsets, independently on how the Variables are defined in the DPM. For each Recordset, there may be 0 or 1 occurrence of each subtype of Standard Key Component.

- **Row Key**: Identifies the Row Ordinate from a Report Table where the selected Variable is located. Arises in Variable Set Selections, when more than one Row for one Report Table is selected. The name for the component is “r”. It is defined on the String Data Type.

- **Column Key**: Identifies the Column Ordinate from a Report Table where the selected Variable is located. Arises in Variable Set Selections, when more than one Column for one Report Table is selected. The name for the component is “c”. It is defined on the String Data Type.

- **Sheet Key**: Identifies the Sheet Ordinate from a Report Table where the selected Variable is located. Arises in Variable Set Selections, when more than one Sheet for one Report Table is selected. The name for the component is “s”. It is defined on the String Data Type.

DPM Key Components are specific to how data is defined in the DPM. Arise when Open Variables are selected, and a Recordset will have one DPM Key Component per each Key Variable associated to the selected Variables.

The name for the DPM Key Components is the Code of the Property associated to the DPM Key Variable.
{tF_32.01, r0020-0040, (c0010, c0030, c0035)}
### F 20.05.a - Geographical breakdown of off-balance sheet exposures by residence of the counterparty (a)

<table>
<thead>
<tr>
<th>Rows</th>
<th>Columns</th>
<th>Nominal amount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ES</td>
<td>0010</td>
</tr>
<tr>
<td>Loan commitments given</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>Financial guarantees given</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>Other commitments given</td>
<td></td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>PT</td>
<td>0010</td>
</tr>
<tr>
<td>Loan commitments given</td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>Financial guarantees given</td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>Other commitments given</td>
<td></td>
<td>600</td>
</tr>
</tbody>
</table>

\{tF_20.05, r0020-0030, c0010\}

<table>
<thead>
<tr>
<th>RCP</th>
<th>r</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES</td>
<td>0020</td>
<td>200</td>
</tr>
<tr>
<td>ES</td>
<td>0030</td>
<td>300</td>
</tr>
<tr>
<td>PT</td>
<td>0020</td>
<td>500</td>
</tr>
<tr>
<td>PT</td>
<td>0030</td>
<td>600</td>
</tr>
</tbody>
</table>
**F 40.01 - Scope of the group: “entity-by-entity”**

<table>
<thead>
<tr>
<th>Rows</th>
<th>Investee</th>
<th>Code</th>
<th>Type of code</th>
<th>Entity name</th>
</tr>
</thead>
<tbody>
<tr>
<td>999</td>
<td></td>
<td>0011</td>
<td>0015</td>
<td>0031</td>
</tr>
<tr>
<td></td>
<td></td>
<td>123456</td>
<td>LEI</td>
<td>Name1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>123456</td>
<td>ISIN</td>
<td>Name2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1111</td>
<td>LEI</td>
<td>Name3</td>
</tr>
</tbody>
</table>

LIN <Key value>   TYC <Key value>

{tF_40.01, c0031}
Outline

• Introduction
• An example
• Information Model
• Main operators
• Changes to current operators
• DPM-ML metamodel
• Calculations
Operators

Selection

- Selection operator
  - With

Arithmetic

- Selection operator
  - Add: (+)
  - Subtract: (-)
  - Multiply: (*)
  - Divide: (/)
  - Absolute value: (abs)
  - Exponential: (exp)
  - Natural logarithm: (ln)
  - Power: (power)
  - Logarithm: (log)
  - Square root: (sqrt)

Comparison

- Equal to: (=)
  - Not equal to: (<>)
  - Greater than: (>)
  - Less than: (<)
  - Element of: (in)
  - Math characters: (match)
  - Is null: (isnull)

Logical operators

- Conjunction: (and)
  - Disjunction: (or)
  - Exclusive disjunction: (xor)
  - Negation: (not)

Aggregate operators

- Sum: (sum)
  - Count: (count)
  - Minimum value: (min)
  - Maximum value: (max)
  - Average: (avg)
  - Median value: (median)

Conditional operators

- If then else: (if)
  - Null substitute: (nvl)
  - Filter: (filter)

String operators

- Length: (len)
  - Concatenate: (&)

Time operators

- Time shift: (time_shift)

Clause operators

- Where: (where)
  - Rename: (rename)
  - Get: (get)
The selection operator

- Curly brackets ({} are used as symbol for the selection operator

- The selection operator has three parts:
  - **Recordset selection**: By referencing:
    - Cells (table, rows, columns and/or sheets).
    - Variables: References to variable codes.
    - Operations: References to the results of other operations.
  - **Default value**: Sets a default value in case the selection has missing data or explicit nulls for a data instance.
  - **Interval**: For numeric variables, selects whether the data should be considered as interval or point.

- The Recordset resulting from a selection is composed by all possible Records in the selection. If one Variable is not reported (missing), then the Record will have null value.
The selection clause – Example

F 01.01 - Balance Sheet Statement [Statement of Financial Position]: Assets

<table>
<thead>
<tr>
<th>Rows</th>
<th>Columns</th>
<th>Rows</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash, cash balances at central banks</td>
<td>Carrying amount</td>
<td>Cash, cash balances at central banks</td>
<td>Carrying amount</td>
</tr>
<tr>
<td>and other demand deposits</td>
<td>0010</td>
<td>and other demand deposits</td>
<td>0010</td>
</tr>
<tr>
<td>Cash on hand</td>
<td>0020</td>
<td>Cash balances at central banks</td>
<td>0030</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>Other demand deposits</td>
<td>0040</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

{tF_01.01, r0010-0040, c0010}

{tF_01.01, r0010-0040, c0010, default:0}

{tF_01.01, r0010-0040, c0010, default:0, interval:true}
The with clause

- The with clause serves to provide a common context to all the selections of an expression.

- In the current Excel files, represented in separate columns.

- The with clause uses the following syntax:

  ```
  with partial_selection: expression
  ```

  - **partial_selection**: Is the selection that is completing the selections in the expression, using the selection clause.

  - **expression**: It is an expression containing selection operators.

- The with clause does not produce an output, but modifies the selections in the expression according to some rules. The operator does not produce a node in DPM-ML.

- The selection in the with applies to all selections in the expression unless they are overridden.
The with clause – Examples

with \{tF\_01.01, c0010, default:0, interval:false\}:
\{r0010\} = \{r0020\} + \{r0030\} + \{r0040\}

No operand in the expression overrides the with context.

with \{tF\_01.01, c0010, default:0, interval:false\}:
\{r0010\} + \{r0040\} = \{tF\_04.01, r0010, c0010\}

The third operand in the expression overrides the table and the column of the with context.

with \{tF\_01.01, c0010, default:0, interval:false\}:
\{tF\_01.01, r0010\} + \{tF\_01.01, r0040\} = \{tF\_04.01, r0010, c0010, default:NULL\}

All three operands in the expression override the table in the context. The third operand overrides also the column and the default.

with \{c0010, default:0, interval:false\}:
\{tF\_01.01, r0010\} + \{tF\_01.01, r0040\} = \{tF\_04.01, r0010\}

No operand in the expression overrides the with context.
Example 1

\[ \{ts.26.01, r0600, (c0060, c0080)\} \]

\[ 0.25 * \{ts.26.01, r0600, (c0060, c0080)\} \]

<table>
<thead>
<tr>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0060</td>
<td>100</td>
</tr>
<tr>
<td>0080</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0060</td>
<td>25</td>
</tr>
<tr>
<td>0080</td>
<td>50</td>
</tr>
</tbody>
</table>
General behaviour for binary operators – Example 2

\[ \{tF_04.02.01, r0120, c0010-0020\} \]

\[
\begin{array}{|c|c|}
\hline
\text{c} & \text{f} \\
\hline
0010 & 100 \\
0020 & 200 \\
\hline
\end{array}
\]

\[ \{tF_04.02.01, r0140, c0010-0020\} \]

\[
\begin{array}{|c|c|}
\hline
\text{c} & \text{f} \\
\hline
0010 & 300 \\
0020 & 400 \\
\hline
\end{array}
\]

with \( \{tF_04.02.01, c0010-0020\} \): \( \{r0120\} + \{r0140\} \)

\[
\begin{array}{|c|c|}
\hline
\text{c} & \text{f} \\
\hline
0010 & 400 \\
0020 & 600 \\
\hline
\end{array}
\]
General behaviour for binary operators – Example 3

\{tc\_28.00, c040\} + \{tc\_28.00, c190\}

**Example 3**

\[
\begin{array}{|c|c|}
\hline
\text{INC} & \text{f} \\
\hline
123 & 1000 \\
456 & 2000 \\
789 & 3000 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{INC} & \text{f} \\
\hline
123 & -100 \\
456 & -200 \\
789 & -300 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{INC} & \text{f} \\
\hline
123 & 900 \\
456 & 1800 \\
789 & 2700 \\
\hline
\end{array}
\]
General behaviour for binary operators – Example 4

\{tF_{40.01}, c0110\} \rightarrow \{tF_{40.02}, c0060\}

\{tF_{40.01}, c0110\} \geq \{tF_{40.02}, c0060\}
General behaviour for binary operators

- If the two *Operands* of a binary *Operator* are * Scalars*, the result shall be the *Scalar* resulting of applying the *Operator* to the *Operands*.

- A binary *Operator* applied to a *Recordset* *Operand* and a *Scalar*, will result in a *Recordset* with the same *Structure* as the input *Recordset* *Operand*. The operator shall be applied to every record of the input *Recordset* and the *Scalar*.

- For two *Recordsets*:
  
  **Constraints**: Binary *Operators* can only be applied to two *Recordsets Operands* if they have:

  - Exactly the same *Key Components*; or
  
  - the *Key Components* of one *Recordset* (Reference *Recordset*) are a superset of the *Key Components* of the other *Recordset*.

  **Behaviour**: Performs an inner join and the operator applies to the pairs of values resulting from performing an inner join.
Harmonising existence check

- Currently there are different ways to check existence. Examples:
  - EIOPA BV195: `not(empty({S.01.02, c0010}))`
  - EBA v6215_m: `{P 01.03} != empty`

- Proposal to change to `isnull()` operator
Harmonising operator symbols

For each operator, only one symbol is allowed. Currently for some operators several symbols are used. Examples:

- $<>$, $!=$
- $in$, $\in$
- $=$, $\equiv$, $==$  
- *Like*, *matches*
Currently there are different practices:

Solution: Parameter in the selection with the default value.

\{tF_01.01, r0010-0040, c0010, default:0\}
Interval applicability in selection

Currently there are different practices:

**EBA**

Currently there are different practices:

**EIOPA**

Solution: Parameter in the selection with the treatment for intervals.

\{tF\_01.01, r0010-0040, c0010, default:0, interval:true\}
Currently there are different practices:

- **EBA**: *where* operator inside *sum* operator
  \[
  \text{sum} \left( \text{where} \left( \{ c \ 08.02, c0010, s0013 \} = 1 \right) \ \{ c \ 08.02, c0020, s0013 \} \right)
  \]

- **EIOPA**: Field in the Excel specifying the filter
  ![Excel Filter Example](image-url)
Filter operator with two arguments:

- The filtered selection
- The condition for the selection

\[
\text{sum(}
\begin{align*}
\text{where(}\{&c 08.02,c0010,s0013\}=1) \\
&c 08.02,c0020, s0013
\end{align*}
\text{)}
\]

\[
\text{sum(}
\begin{align*}
\text{filter(} & \{tc 08.02, c0020, s0013\}, \\
&\{tc 08.02, c0010, s0013\} = 1
\end{align*}
\text{)}
\]

\[
\text{isnull(}
\begin{align*}
\text{filter(} & \{c0330\}, \\
&\text{match(}\{c0320\}, ‘##D3’)
\end{align*}
\text{)}
\]

\[
\begin{array}{ll}
\text{Filter} & \text{Validation} \\
\{c0320\} & \text{like ‘##D3’} \\
\{c0330\} & \text{=} \text{empty}
\end{array}
\]
Currently EBA and EIOPA use columns in the Excel to define the group by:

**EBA:**

<table>
<thead>
<tr>
<th>ID</th>
<th>sheets</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>v10666_m</td>
<td>(All)</td>
<td>(C 08.01.a, r0070, c0240) * sum((C 08.02, c0140, (rNNN)) = sum((C 08.02, c0240, (rNNN)) * (C 08.02, c0140, (rNNN)))</td>
</tr>
</tbody>
</table>

**EIOPA:**

<table>
<thead>
<tr>
<th>Validation ID</th>
<th>Template 1</th>
<th>Columns</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BV35</td>
<td>S.16.01</td>
<td>c0020-0080</td>
<td>{r0200}=sum({(r0040-0190)})</td>
</tr>
</tbody>
</table>
Aggregate operators have an optional argument which is a list of the Key Components for the group by.

\[
\{c \ 08.01.a, \ r0070, \ c0240\} \times \sum(\{C \ 08.02, \ c0140\}) = \sum(\{c \ 08.02, \ c0240\} \times \{C \ 08.02, \ c0140\})
\]
Other changes

- **with** clause: Added to deal with the scope.

- **xsum** operator: Deleted because it is not necessary under the new approach.

- **string_length** operator: Renamed to `len`, to make it more consistent with other languages (e.g., Excel).

- **Regex** harmonisation: One Regex flavour selected.
Outline

Introduction

An example

Information Model

Main operators

Changes to current operators

DPM-ML metamodel

Calculations
Operations have a code and can be grouped.

Groups of operations refer to the cases where users define a parent validation and the system derives automatically children validations. There can only be two levels of operations (i.e., a child validation cannot be the parent of another validation). The types of validations that generate children are Variant and Property constraint validations.

Operations can have many versions. A new version of an operation is required whenever any of their attributes change.
\[
\{F\ 36.02.c, (r0140, r0210), c0010\} = \\
\text{sum(} \\
\{F\ 32.02.a, (r0140, r0210), (c0040, c0070)\} * \\
\{F\ 32.02.b, (r0140, r0210), (c0040, c0070)\} \\
\text{group by } r) \\
\]
Operation tree: Example xyz index

\[
\{tF_{.02.a}, (r0140, r0210), (c0040, c0070)\} * \\
\{tF_{.02.b}, (r0140, r0210), (c0040, c0070)\}
\]

\[
\sum( \\
\{tF_{.02.a}, (r0140, r0210), (c0040, c0070)\} * \\
\{tF_{.02.b}, (r0140, r0210), (c0040, c0070)\}
\)
\] group by r

\[
\{tF_{.02.c}, (r0140, r0210), (c0010)\} = \\
\sum( \\
\{tF_{.02.a}, (r0140, r0210), (c0040, c0070)\} * \\
\{tF_{.02.b}, (r0140, r0210), (c0040, c0070)\}
\)
\] group by r

\[
\begin{align*}
\{tF_{.02.c}, (r0140, r0210), (c0010)\} &= \\
\sum( \\
\{tF_{.02.a}, (r0140, r0210), (c0040, c0070)\} * \\
\{tF_{.02.b}, (r0140, r0210), (c0040, c0070)\}
\) \text{ group by } r
\end{align*}
\]
Each operation version can be applied in different scopes. An operation scope refers to the individual module versions or sets of module versions (for cross-module validations) to which the operation logically applies.

For each operation scope, there may be different values for the attributes:

- **IsActive**: Determines whether the operation shall be run for a certain scope.
- **Severity**: Determines the severity of the error, if the validation is not passed.
- **FromSubmissionDate**: Sets a date from which the validation applies to subsequent submissions.
For calculations (i.e., operations that serve to calculate values for variables), it is necessary to link the operation to the actual variable that is generated. This is done with the **VariableCalculation** table, which links to the **Variable** and the **Module**. The link to the **Module** is necessary due to the fact that a **Variable** can be used in several **Modules**, being calculated in some cases and not calculated in other cases.
Outline

- Introduction
- An example
- Information Model
- Main operators
- Changes to current operators
- DPM-ML metamodel
- Calculations
About calculations

- Calculations are very similar to validations: **Algorithms** to manipulate input data into output data.

- The DPM-XL and DPM-ML for validations can also be used for calculations.

- Only difference is in the treatment of related issues, like triggers to run validations vs calculations, or what to do with the results of execution.
Consider the current transparency exercise table with mapping:

<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
<th>OWN FUNDS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>2</td>
<td>COMMON EQUITY TIER 1 CAPITAL (net of deductions and after applying transitional adjustments)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A.1.1</td>
<td>3</td>
<td>Capital instruments eligible as CET1 Capital (including share premium and net own capital instruments)</td>
</tr>
<tr>
<td>A.1.2</td>
<td>4</td>
<td>Retained earnings</td>
</tr>
<tr>
<td>A.1.3</td>
<td>5</td>
<td>Accumulated other comprehensive income</td>
</tr>
<tr>
<td>A.1.4</td>
<td>6</td>
<td>Other Reserves</td>
</tr>
<tr>
<td>A.1.5</td>
<td>7</td>
<td>Funds for general banking risk</td>
</tr>
<tr>
<td>A.1.6</td>
<td>8</td>
<td>Minority interest given recognition in CET1 capital</td>
</tr>
<tr>
<td>A.1.7</td>
<td>9</td>
<td>Adjustments to CET1 due to prudential filters</td>
</tr>
<tr>
<td>A.1.8</td>
<td>10</td>
<td>(-) Intangible assets (including Goodwill)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>As of 30/09/2020</th>
<th>As of 31/12/2020</th>
<th>As of 31/03/2021</th>
<th>As of 30/06/2021</th>
<th>COREP CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>C 01.00 (r010,d010)</td>
</tr>
<tr>
<td>2</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>#VALUE!</td>
<td>C 01.00 (r020,d010)</td>
</tr>
<tr>
<td>3</td>
<td>C 01.00_030_010</td>
<td>C 01.00_030_010</td>
<td>C 01.00_030_010</td>
<td>C 01.00_00030_00010</td>
<td>C 01.00 (r030,d010)</td>
</tr>
<tr>
<td>4</td>
<td>C 01.00_0130_010</td>
<td>C 01.00_0130_010</td>
<td>C 01.00_0130_010</td>
<td>C 01.00_00130_00010</td>
<td>C 01.00 (r130,d010)</td>
</tr>
<tr>
<td>5</td>
<td>C 01.00_180_010</td>
<td>C 01.00_180_010</td>
<td>C 01.00_180_010</td>
<td>C 01.00_00180_00010</td>
<td>C 01.00 (r180,d010)</td>
</tr>
<tr>
<td>6</td>
<td>C 01.00_200_010</td>
<td>C 01.00_200_010</td>
<td>C 01.00_200_010</td>
<td>C 01.00_00200_00010</td>
<td>C 01.00 (r200,d010)</td>
</tr>
<tr>
<td>7</td>
<td>C 01.00_0210_010</td>
<td>C 01.00_0210_010</td>
<td>C 01.00_0210_010</td>
<td>C 01.00_00210_00010</td>
<td>C 01.00 (r210,d010)</td>
</tr>
<tr>
<td>8</td>
<td>C 01.00_0230_010</td>
<td>C 01.00_0230_010</td>
<td>C 01.00_0230_010</td>
<td>C 01.00_00230_00010</td>
<td>C 01.00 (r230,d010)</td>
</tr>
<tr>
<td>9</td>
<td>C 01.00_0250_010</td>
<td>C 01.00_0250_010</td>
<td>C 01.00_0250_010</td>
<td>C 01.00_00250_00010</td>
<td>C 01.00 (r250,d010)</td>
</tr>
<tr>
<td>10</td>
<td>C 01.00_0300_010+C 01.00_0340_010</td>
<td>C 01.00_0300_010+C 01.00_0340_010</td>
<td>C 01.00_0300_010+C 01.00_0340_010</td>
<td>C 01.00_00300_00010+C 01.00_00340_00010</td>
<td>C 01.00 (r300,d010)+ C 01.00 (r340,d010)</td>
</tr>
</tbody>
</table>
The table could be represented as a DPM table, and each cell would be a variable.

The DPM-XL (and the related metadata) can assign the results of operations to DPM variables.

For calculations, being able to concatenate operations is critical.

**Example: calculation of variables in row 10 (DPM-XL):**

\[
\text{TE\_capital\_r10} := \{\text{tc\_01.00, r0300, c0010}\} + \{\text{tc\_01.00, r0410, c0010}\};
\]

\[
\text{tTE\_capital\_r10\_c1} \leftarrow \\
\quad \text{filter(oTE\_capital\_r10, date='2020-09-30')};
\]

\[
\text{tTE\_capital\_r10\_c2} \leftarrow \\
\quad \text{filter(oTE\_capital\_r10, date='2020-12-31')};
\]

...