Data management aspects in the context Gaia's distributed processing

Jose Hernandez*, Alexander Hutton*, Hassan Siddiqui*

* European Space Astronomy Centre of ESA, Madrid, Spain

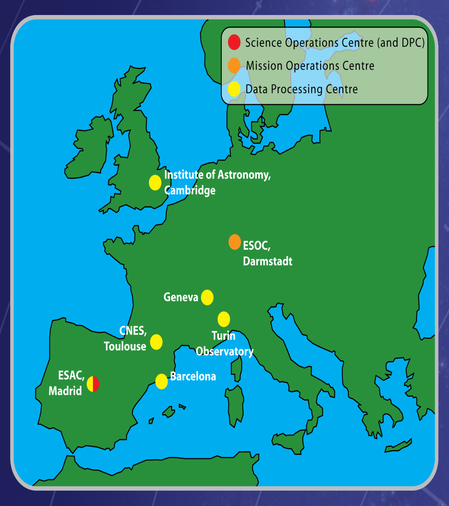
Abstract

Gaia is ESA's ambitious space astrometry mission with a foreseen launch date around mid-2013. Its main objective is to perform a stellar census of the 1000 million brightest objects in our galaxy from which an astrometric catalog of micro-arcsec level accuracy will be constructed. The reduction of Gaia's data is a convoluted process involving several European data centers distributed across Europe. Data will be shipped to these centers where part of the processing will take place and a central database will collect and integrate the results of the data reduction.

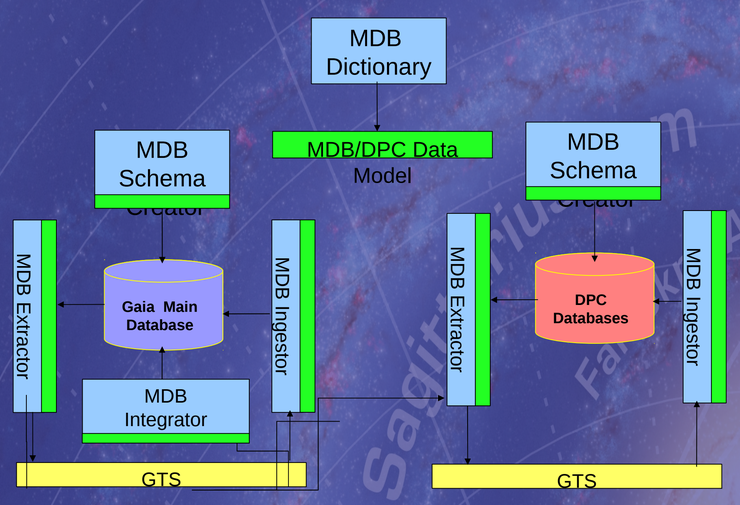
At past ADASS meetings we have presented the tools developed to support Collaborative Data Modeling in the distributed context of Gaia. Ensuring that the right version of the data gets used at the right time and place will be far from trivial. We will describe how the data will be versioned and what will be the mechanisms put in place in order to track its provenance and control how the updated data gets used at the processing centers.

Gaia Main Database

- Gaia Main Database will be the central repository of Gaia's Data located at ESAC
- It will receive and redistribute the Data to 5 different processing Centers:
  - Barcelona: Intermediate Data Update
  - Cambridge: Photometric Processing
  - Geneva: Variability Analysis
  - CNES-Toulouse: Spectroscopy, Object Processing, Classification
  - Torino: Verification



Data Processing Centers



- The Data reduction will be an iterative process in order to deal with the interdependencies.
- There will be several versions of the Main Database with increasing volumes of data
- Some data will be generated and distributed continuously as it is received from the S/C and processed
- Other types of data will be available in one go at the end of the Processing Cycle

Data Qualification

- We need to have the capability to send new versions of the same data within a processing cycle
- This will happen when some problem is found in the pipelines and data needs to be regenerated
- It may also happen when we get a more accurate version of some data (like Gaia's orbit)
- The MDB will issue a message to the Data Processing Centers “qualifying” some data
- Upon reception of the Qualifying messages the DPCs will execute them on the local Stores

```
<xs:element name="DataQualifier">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="qualifierDpc" type="dpc" maxOccurs="1" minOccurs="1"/>
            <xs:element name="id" type="xs:long" maxOccurs="1" minOccurs="1"/>
            <xs:element name="generationTime" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="status" type="status"  maxOccurs="1" minOccurs="1"/>
            <xs:element name="qualifiedData" type="dataSet"  minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:complexType name="dataSet">
    <xs:sequence>
        <xs:element name="dataSelectExpr" type="xs:string"
            maxOccurs="1" minOccurs="1" />
        <xs:element name="dataTypes" type="xs:string"
            maxOccurs="1" minOccurs="1" />
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="status">
    <xs:restriction base="xs:string">
        <xs:enumeration value="VALID"/>
        <xs:enumeration value="RETRACTED"/>
        <xs:enumeration value="CORRUPTED"/>
        <xs:enumeration value="VALID_OUTDATED"/>
    </xs:restriction>
</xs:simpleType>
```

Versioning the Data and Tracking its Origin

- Gaia's reduced data will be tagged with a solutionId
- This is a long number which encodes some information:
  - What software generates the data
  - When was the data generated
  - A counter of the executions of a given software

64 bits long				
1 bit	10 bits	11 bits	15 bits	27 bits
	SW Identifier	SW version	Day number	executionId

- Each record in Gaia's data has a field of type solutionId
- Each solutionId has is a corresponding SolutionIdSummary Object giving more information
- The solutionId encoding has some interesting properties:
  - No collision between ranges used by different systems
  - They increase monotonically with time
  - They have enough range to cover all the executions of the SW
- The solutionId also allows the Data Processing Centers to manage different versions of the same data on a single store and select the appropriate one
- The data size should not be an issue when the data is compressed
- Each Data Processing centre will maintain a master database tracking the solutionIds already consumed and assigning new ones
- The solutionId is generated automatically by inspecting the Meta-Data from the SW jar file being executed

Some Scenarios

- Data Provenance will also be tracked
- For each solutionId the data types and solutionIds used as input will be recorded
- The real input data used will be validated against the “expected” input data
- This requires the DPCs to perform some book-keeping

Some possible scenarios

- DPCx realizes that some of the data sent to the MDB needs to be retracted and shouldn't be used
- DPCx generates a qualifying message for the relevant MDB tables and solutionIds and sends them to the MDB
- MDB will resend the received qualifying message to all the DPCs

Upon reception of the DataQualifier message at the DPC there will be different possibilities:

- DPCy checks that it doesn't use the retracted data, it doesn't need to take any action
- DPCz uses the retracted data which had already been ingested locally at DPCz store but not processed yet, DPCz identifies the ingested data by the solutionId and the MDB type and deletes it
- DPCw uses the retracted data which had already been ingested locally at DPCw, this data was already processed. Here we could have two scenarios:

- DPCw stops the processing, deletes the retracted data and also the corresponding output of the DPCw pipeline
- DPCw can't reprocess the data, instead it flags as invalid some of its pipeline output affected by the data

Data management aspects in the context Gaia's distributed processing

Jose Hernandez*, Alexander Hutton*, Hassan Siddiqui*
* European Space Astronomy Centre of ESA, Madrid, Spain

Abstract

Gaia is ESA's ambitious space astrometry mission with a foreseen launch date around mid-2013. Its main objective is to perform a stellar census of the
1000 million brightest objects in our galaxy from which an astrometric catalog of unprecedented level accuracy will be constructed. The reduction of Gaia's data is a coordinated process involving several European
data centers distributed across Europe. Data will be shipped to these centers where part of the processing will take place and a central database will collect and integrate the results of the data reduction.

At past ADASS meetings we have presented the tools developed to support collaborative Data Modeling in the distributed context of Gaia. Ensuring that the right version of the data gets used at the right
time and place will be the main critical. We will describe how the data will be versioned and what will be the mechanisms put in place in order to track its provenance and control how the updated data gets used at
the processing centers.

Gaia Main Database

- Gaia Main Database will be the central repository of Gaia's Data located at ESAC
- It will receive and redistribute the Data to 5 different processing Centers:
  - Barcelona: Intermediate Data Update
  - Cambridge: Photometric Processing
  - Geneva: Variability Analysis
  - CNES Toulouse: Spectroscopy, Object Processing, Classification
  - Torino: Verification

- The Data reduction will be an iterative process in order to deal with the interdependences.
- There will be several versions of the Main Database with increasing volumes of data.
- Some data will be generated and distributed continuously as it is received from the S/C and processed
- Other types of data will be available in one go at the end of the Processing Cycle

Data Qualification

- We need to have the capability to send new versions of the same data within a processing cycle
- This will happen when some problem is found in the pipelines and data needs to be reprocessed
- It may also happen when we get a more accurate version of some data (like Gaia's orbit)
- The MDB will issue a message to the Data Processing Centers "qualifying" some data
- Upon reception of the Qualifying messages the DPCs will execute them on the local Stores

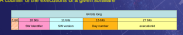
```
<xs:element name="DataQualifier">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="qualifiedOn" type="file" minOccurs="1" maxOccurs="1"/>
      <xs:element name="dr" type="string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="generationTime" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="dataType" type="value" minOccurs="1" maxOccurs="1"/>
      <xs:element name="qualifiedData" type="dataSet" minOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="dataSet">
  <xs:sequence>
    <xs:element name="dataSetInTime" type="xs:string"
                minOccurs="1" maxOccurs="1"/>
    <xs:element name="dataType" type="xs:string"
                minOccurs="1" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="data">
  <xs:restriction base="xs:string">
    <xs:enumeration value="VALID"/>
    <xs:enumeration value="RETRACTED"/>
    <xs:enumeration value="SUBSTITUTED"/>
    <xs:enumeration value="VALID_OUTDATED"/>
  </xs:restriction>
</xs:simpleType>
```

Versioning the Data and Tracking its Origin

- Gaia's reduced data will be tagged with a solutionId
- This is a long number which encodes some information:
  - What software generates the data
  - When was the data generated
  - A counter of the executions of a given software



- Each record in Gaia's data has a field of type solutionId
- Each solutionId has is a corresponding SolutionIdSummary Object giving more information
- The solutionId encoding has some interesting properties:
  - No collision between crops used by different systems
  - They increase monotonically with time
  - They have enough range to cover all the executions of the SW
- The solutionId also allows the Data Processing Centers to manage different versions of the same data on a single store and select the appropriate one
- The data size should not be an issue when the data is compressed
- Each Data Processing centre will maintain a master database tracking the solutionids already consumed and assigning new ones
- The solutionId is generated automatically by inspecting the Meta-Data from the SW jar file being executed

Some Scenarios

- Data Provenance will also be tracked
- For each solutionId the data types and solutionIds used as input will be recorded
- The real input data used will be validated against the "expected" input data
- This requires the DPCs to perform some book-keeping

Some possible scenarios

- DPCs realizes that some of the data sent to the MDB needs to be retracted and shouldn't be used
- DPCs generates a qualifying message for the relevant MDB tables and solutionids and sends them to the MDB
- MDB will resend the received qualifying message to all the DPCs

Upon reception of the DataQualifier message at the DPC there will be different possibilities:

- DPCs checks that it doesn't use the retracted data, it doesn't need to take any action
- DPCs uses the retracted data which had already been ingested locally at DPCs store but not processed yet, DPCs identifies the ingested data by the solutionId and the MDB type and deletes it
- DPCs uses the retracted data which had already been ingested locally at DPCs, this data was already processed. Here we could have two scenarios:

- DPCs stops the processing, deletes the retracted data and also the corresponding output of the DPCs pipeline
- DPCs can't reprocess the data, instead it flags an invalid some of its pipeline output affected by the data