CORE – COmmon Reference Environment

Donato Summa
Monica Scannapieco
Antonino Virgillito

Istat
Outline

- Introduction
- CORE Design
- CORE Architectural Components
- Illustration of CORE Platform
- Case studies
- CORE Follow-up
Introduction
CORE Generalities

- Principal Outcome: Environment for the definition and execution of standard statistical processes
  - Definition of a process in terms of available services
  - Execution of the composed workflow
CORE Generalities

“Plug and play” approach to process execution
CORE Generalities

“Plug and play” approach to process execution
Why CORE?

START

Allocation (MAUSS – R)

Selection (SAS Script)

Estimation (ReGenesees)

STOP
Why CORE?

START

Allocation (MAUSS – R)

Selection (SAS Script)

Estimation (ReGenesees)

STOP
Why CORE?

Technological Heterogeneity

- Different technologies
- Different formats
- ...

Allocation (MAUSS – R)

Selection (SAS Script)

Estimation (ReGenesees)

STOP
Why CORE?

Technological Heterogeneity
- Different technologies
- Different formats
- ...

Data Heterogeneity
- Different names for variables
- Variables as combinations of other variables
- ...

Allocation (MAUSS)

Selection (SAS Script)

STOP
Why CORE?

• Technological heterogeneity can be solved by solutions available on the market

CORE permits to solve both technological and data heterogeneity in a single environment
CORE Vision

1. **Abstract services**: *well-defined, technology-independent* functionalities implemented by different IT tools;
2. **Statistical process**: workflow defined in terms of available services;
3. **Data model**: *standardization* of the semantics/format of services data, i.e. definition of the domain entities involved as input/output between services.
CORE Vision

1. Abstract services: well-defined, technology-independent functionalities implemented by IT tools

- Allocation (MAUSS – R)
- Selection (SAS Script)
- Estimation (ReGenesees)
CORE Vision

3. Data model: *standardization* of the semantics/format of services data

3.1 Domain descriptor (DD)

```xml
<schema name="DEMO_DD">
  <entity name="SamplePlan">
    <property name="VAR"/>
    <property name="SIZE"/>
  </entity>
  ...  
</schema>
```

3.2 Mapping to/from DD
CORE Design Tasks - 1

- Design of services
- Definition of integration APIs (IAPIs)
- Data conversion from/to CORA model to/from tool specific format
- Graphical front ends for designing schemas and mappings
CORE Design Tasks - 2

- Design of processes
- How to define and execute processes within CORE
  - Modelling language
  - Execution
  - Visual interfaces design
- Design of a service repository
CORE Design Tasks - 3

• Design of exchanged data
• Definition of data models and formats (plain XML/XSD, SDMX...) to be used for data exchanges
• Definition of metadata necessary for process execution
• SDMX Relationships
CORE Design
CORE Design: Services

- Abstract services: specify a well-defined functionality in a technology-independent way
- An abstract service can be implemented by one or more concrete services, i.e. IT tools
- Examples: sample allocation, record linkage, estimates and errors computation, etc.
CORE Design: Services

- **GSBPM classification**
  - **Documentation purpose**
  - Provided that a CORE service can be linked to IT tools, GSBPM tagging enables the performance of a search e.g. retrieving
    
    “all the IT tools implementing the 5.4 Impute subprocess of GSBPM proposal”
CORE Design: Services

- Service inputs and outputs
  - Specified by logical names
  - Characterized with respect to their “role” in data exchange

**Non-CORE**: if they are not provided by/to other services of the process, but are only “local” to a specific service

**CORE**: they are passed by/to other services and hence they do need to undergo CORE transformations
CORE Design: Data and Metadata

- They are specified as service inputs and outputs
  - Logical names link them to previously specified services
  - Non-CORE data only need the file system path where they can be retrieved
CORE Design: CORE Data

- The specification of CORE data is provided by 3 elements:
  - Domain descriptor
  - CORE data model
  - Mapping model
Domain Descriptor: Model

- Entity
  - Like “entities” in Entity Relationships
- Entity properties
  - Like “attributes” in Entity Relationships
- Very simple (meta-)model
<schema name="DEMO_Domain_Descriptor">
<entity name="SamplePlan">
   <property name="STRATIFICATION_VAR"/>
   <property name="STRATUM_SAMPLE_SIZE"/>
   <property name="STRATUM_POPULATION_SIZE"/>
</entity>
<entity name="Enterprise">
   <property name="IDENTIFIER"/>
   <property name="STRATIFICATION_VAR"/>
   <property name="WEIGHT"/>
   <property name="SAMPLING_FRACTION"/>
   <property name="ENTERPRISE_FLAG"/>
   <property name="EMPLOYEES_NUM"/>
   <property name="VALUE_ADDED"/>
   <property name="AREA"/>
</entity>
</schema>
Domain Descriptor Role

- Role of the Domain Descriptor (DD): from service-to-service data mapping to service-to-global data mapping

\[ \begin{align*}
\text{S}_1 & \xrightarrow{i_1} \text{DD} \\
\text{S}_1 & \xrightarrow{o_1} i_1 \\
\text{S}_2 & \xrightarrow{i_2} \text{DD} \\
\text{S}_2 & \xrightarrow{o_2} i_2 \\
on_1 & \quad \quad \rightarrow \quad \quad i_2 \\
\text{O}_1 \text{ mapped to } i_2 \\
\text{Via ad-hoc mapping} \\
\text{O}_1 & \quad \quad \rightarrow \quad \quad \text{DD} \\
\text{O}_1 \text{ mapped to } i_2 \\
\text{Via DD} \\
\end{align*} \]
CORE Data Model: Role

- Specified once and valid for all processes
- Extensible, i.e. core tag, data set kind, column kind can be modified
- Adds more semantics to data
  - Example of usage: mapping to other models
CORE Data Model

- Rectangular data set
- CORE tag:
  - Data set level (mandatory)
  - Column level (optional)
  - Rows level (optional)
- Data set kind
- Column kind
CORE Data Model Role

• Specified once and valid for all processes
• Extensible, i.e. core tag, data set kind, column kind can be modified
• Adds more semantics to data
  • Example of usage: mapping to other models
Mapping Model

- Rectangular data assumption
- Mapping is intended to be specified with respect to Domain Descriptor
  - Columns are to be mapped to properties of an entity
- It contains the specification of how CORE data model concepts are associated to data
CORE Logical Architecture
CORE GUIs

- Process design
  - **Service data flow**
- Service design
  - **Set of interfaces for the definition of services and related data flow**
- Data design
  - **Set of interfaces for the specification of domain descriptors and mapping files**
Use Case Specification

• CORE (Principal) Users
Use Case Specification: Tool Management
Use Case Specification: Service Management

- Statistical User
  - Service Management
    - Add Service
    - Modify Service
    - Delete Service
    - Show Services' List
      - Select Service
      - Select Tool
      - Show Tools' List
Use Case Specification: Process

Diagram:

- Statistical User
- Process Definition
- Show Services' List
- Show Service
- Select Service

Relationships:
- Statistical User uses Process Definition
- Process Definition uses Show Services' List
- Show Services' List uses Show Service
- Show Service uses Select Service
Integration APIs

- Purpose: wrapping a tool by a CORE service
  
  - Translates inputs and outputs of the tool in a completely transparent and automatic way
Repository

- Processes and their instances
- Services with their GSBPM and CORE classifications
- Tools and their runtime features
- Data with their logical classification within CORE processes
Process Engine

- Official statistics processes can be viewed from two perspectives:
  - Functional: they are data-oriented, reflecting a common feature of scientific workflows
  - Organizational: they are workflow-oriented, have the complexity of real production lines, with the need for harmonizing the work of different actors
Process Engine

- Hence our process engine has two layers ...

- **WF ENGINE**
- **DATA FLOW CONTROL SYSTEM**

**Complex control flows**
- ✔ Synchronizing constructs, cycles, conditions, etc.
- ✔ E.g.: Interactive multi-user editing imputation

**Simple control flows**
- ✔ Sequence of tasks is composed by connecting the output of one task to the input of another
- ✔ Data intensive operations
Architecture Deployment

- Web-based architecture centered on a centralized component
  - CORE Environment
- Different CORE deployments can co-exist
  - Intra- or Inter- organization
- Services can be remotely executed
  - Support is needed in the form of a distributed component for tool execution and data transfer
Types of service runtime

- **Batch**
  - Tool executed by a command line call
  - Can be automated

- **Interactive**
  - User interact with the tool through a tool-provided GUI
  - Cannot be automated

- **Web service**
  - No tool – procedure distributed on a web service activated by a programming language call
  - Can be automated
CORE Technical Architecture

CORE Environment

GUI

Definition Repository

Process Engine

Integration APIs

Runtime

Remote activation

Web service client

Runtime

Runtime agent

Batch-Interactive runtime

Run on the machine on which the tool is deployed. Is responsible for:
- Preparing the input
- Gathering the output
- Activating the tool

Web service runtime

Web container

Web service client
CORE Technical Architecture

CORE Environment

- GUI
- Definition Repository
- Process Engine
- Integration APIs
- Runtime
  - Remote activation
  - Web service client

Batch-Interactive runtime

Runtime
- Runtime agent

The process engine signals a service must be executed
Service definition is extracted from the repository, as well as the required datasets and the corresponding mappings.
Datasets are converted according to the mapping.
CORE Technical Architecture

CORE Environment

- **GUI**
- **Definition Repository**
- **Process Engine**
- **Integration APIs**
- **Runtime**
  - Remote activation
  - Web service client

Batch-Interactive runtime

- **Runtime**
  - Runtime agent

Converted datasets are transferred to the remote runtime
The tool is activated by the runtime agent.
The output datasets are gathered and sent back to the CORE environment.
Datasets are converted back to CORE format according to the mapping.
CORE Technical Architecture

CORE Environment

- GUI
- Definition Repository
- Process Engine
- Integration APIs
- Runtime
  - Remote activation
  - Web service client

Batch-Interactive runtime

- Runtime
  - Runtime agent

Converted datasets are stored in the repository

Eurostat
CORE Technical Architecture

- CORE Environment
  - GUI
  - Definition Repository
  - Process Engine
  - Integration APIs
  - Runtime
    - Remote activation
    - Web service client

- Batch-Interactive runtime
  - Runtime
    - Runtime agent

The process continues its execution
Scenario 1

• Remote execution command line/GUI
  – Physical layers: CORE env, Service
  – AGENT
Scenario 2

• Remote execution web service
  – Physical layers: CORE env, Service
CORE Scenario
Why a Process Scenario?

• Helps to clarify ideas and to assess their feasibility
• Forces to make newly proposed solutions concrete
• Can/will be used as empirical test-bed during the whole implementation cycle of the CORE environment
How did we build the Scenario?

• Rationale for our Scenario:
  • **Naturality**: involves typical processing steps performed by NSIs for sample surveys
  • **Minimality**: very easy workflow (no conditionals, nor cycles), can be run without a Workflow Engine
  • **Appropriateness**: incorporates as much heterogeneity as possible: heterogeneity is precisely what CORE must be able to get rid of
Spreading Heterogeneity over the Scenario

• The Scenario incorporates both:
  
  • **Data Heterogeneity**
    Via data exchanged by CORE services belonging to the scenario process
  
  • **Technological Heterogeneity**
    Via IT tools implementing scenario sub-processes
Data Heterogeneity

• The Scenario entails different levels of data heterogeneity:
  • **Format Heterogeneity**: CSV files, relational DB tables, SDMX XML files involved
  • **Statistical Heterogeneity**: both Micro and Aggregated Data involved
  • “Model” Heterogeneity: some data refer to ordinary real-world concepts (e.g. enterprise, individual, …), some other to concepts arising from the statistical domain (e.g. stratum, variance, sampling weight, …)
Technological Heterogeneity

• The Scenario requires to wrap inside CORE-compliant services very different IT tools:
  • simple SQL statements executed on a relational DB
  • batch jobs based on SAS or R scripts
  • full-fledged R-based systems requiring a human-computer interaction through a GUI layer
The Scenario at a glance

**ALLOCATION**
- START
  - Compute Strata Statistics
  - Allocate the Sample
  - Select the Sample

**ESTIMATION**
- Calibrate Survey Data
- Compute Estimates and Sampling Errors
- Store Estimates and Sampling Errors
- Convert to SDMX
- STOP
Sample Allocation Subprocess

- **Overall Goal**: determine the minimum number of units to be sampled inside each stratum, when lower bounds are imposed on the expected level of precision of the estimates the survey has to deliver.
- **Two statistical services are needed**:
  - Compute Strata Statistics
  - Allocate the Sample
Compute Strata Statistics Service

- **Goal**: compute, for each stratum, the population mean and standard deviation of a set of auxiliary variables
- **IT tool**: a simple SQL aggregated query with a group-by clause
  - **NSIs usually maintain their sampling frame(s) as Relational DB tables**
- **Integration API**: must support Relational/CORE transformations
- **CORA tag**: “Statistics”
Allocate the Sample Service

- **Goal**: solve a constrained optimization problem to find and return the optimal sample allocation across strata
- **IT tool**: Istat MAUSS-R system
  - implemented in R and Java, can be run either in batch mode or interactively via a GUI
- **Integration API**: must support CSV/CORE transformations
  - MAUSS handles I/O via CSV files
- **CORA tag**: “Statistics”
Sample Selection Subprocess

- **Goal**: draw a stratified random sample of units from the sampling frame, according to the previously computed optimal allocation
- **IT tool**: a simple SAS script to be executed in batch mode
- **Integration API**: CSV/CORE transformation
  - SAS datasets have proprietary, closed format ⇒ we’ll not support direct SAS/CORE conversions
- **CORA tag**: “Population”
  - output stores the identifiers of the units to be later surveyed + basic information needed to contact them
Estimation Subprocess

- **Overall Goal**: compute the estimates the survey must deliver, and assess their precision as well
- **Two statistical services are needed**:
  - **Calibrate Survey Data**
  - **Compute Estimates and Sampling Errors**
Calibrate Survey Data Service

- **Goal**: provide a new set of weights (the “calibrated weights”) to be used for estimation purposes
- **IT tool**: Istat ReGenesees system
  - implemented in R, can be run either in batch mode or interactively via a GUI
- **Integration API**: can use both CSV/CORE and Relational/CORE transformations
- **CORA tag**: “Variable”
Estimates and Errors Service

- **Goal**: use the calibrated weights to compute the estimates the survey has to provide (typically for different subpopulations of interest) along with the corresponding confidence intervals
- **IT tool**: Istat ReGenesees system
- **Integration API**: can use both CSV/CORE and Relational/CORE transformations
- **CORA tag**: “Statistic”
Store Estimates Subprocess

- **Goal**: persistently store the previously computed survey estimates in a relational DB
  - e.g. in order to subsequently feed a data warehouse for online publication
- **IT tool**: a set of SQL statements
- **Integration API**: Relational/CORE transformation again
- **CORA tag**: “Statistics”
Convert to SDMX Service

• **Goal**: retrieve the aggregated data from the relational DB and directly convert them in SDMX XML format
  
  • *e.g. to later send them to Eurostat*

• **IT tool**: ???

• **Integration API**: must support SDMX/CORE transformations

• **CORA tag**: “Statistics”
Scenario Open Issues

• Besides I/O data, CORE must be able to handle “service behaviour parameters”. How?
  • e.g. to analyze a complex survey, ReGenesees needs a lot of sampling design metadata, namely information about strata, stages, clusters identifiers, sampling weights, calibration models, and so on
• Enabling the CORE environment to support interactive services execution is still a challenging problem
  • we plan to exploit MAUSS-R and/or ReGenesees to test the technical feasibility of any forthcoming solution
• How to implement a SDMX/CORE converter?
Demo Scenario

- Involves 3 typical processing steps performed by NSIs for sample surveys:
  - **Sample Allocation**
  - **Sample Selection**
  - **Estimation**
- It has been used as empirical test-bed during the whole implementation cycle of the CORE environment
Rationale for the Scenario

• **Minimality**: very easy workflow (no conditionals, nor cycles), can be run without a Workflow Engine

• **Appropriateness**: addresses heterogeneity issues
  • *heterogeneity* is precisely what CORE must be able to get rid of
Spreading Heterogeneity over the Scenario

• The Scenario incorporates both:
  • **Data Heterogeneity**: Via data exchanged by CORE services belonging to the scenario process
  • **Technological Heterogeneity**: Via IT tools implementing scenario services
    A batch job based on a SAS script
    Two full-fledged R-based systems
The Scenario at a glance
Sample Allocation Service

- **Overall Goal**: determine the minimum number of units to be sampled inside each stratum, when lower bounds are imposed on the expected level of precision of the estimates the survey has to deliver.

- **IT tool**: Istat MAUSS-R system
  - implemented in R and Java
  - CORA tag: “Statistics”
Sample Selection Service

- **Goal**: draw a stratified random sample of units from the sampling frame, according to the previously computed optimal allocation
- **IT tool**: a simple SAS script to be executed in batch mode
- **CORA tag**: “Population”
Estimates and Errors Service

- **Goal**: compute the estimates the survey has to provide (typically for different subpopulations of interest) along with the corresponding confidence intervals
- **IT tool**: Istat ReGenesees System
  - **R-based**
  - **CORA tag**: “Statistics”