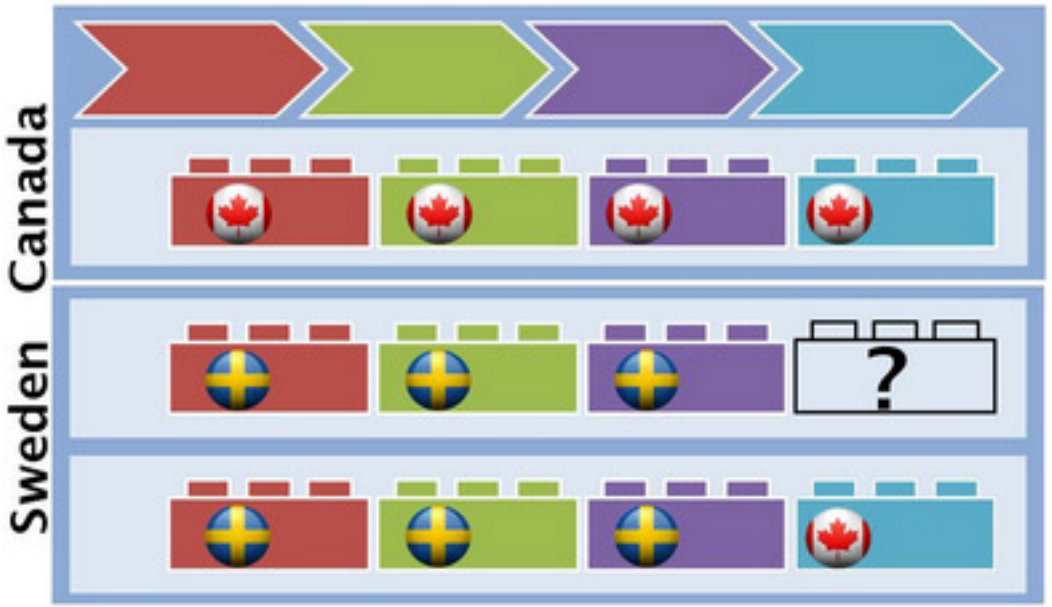
CSPA Use Cases and Case Studies

Antonino Virgillito

Istat

CSPA Use Cases

- Sharing and reusing of statistical software among NSIs



CSPA Proof Of Concept

- The Proof of Concept produced the first CSPA Statistical Services.
- The work was progressed in parallel to the work undertaken to develop the architecture.
- The purpose of doing this was to test the architecture and provide quick feedback into the development of the architecture.

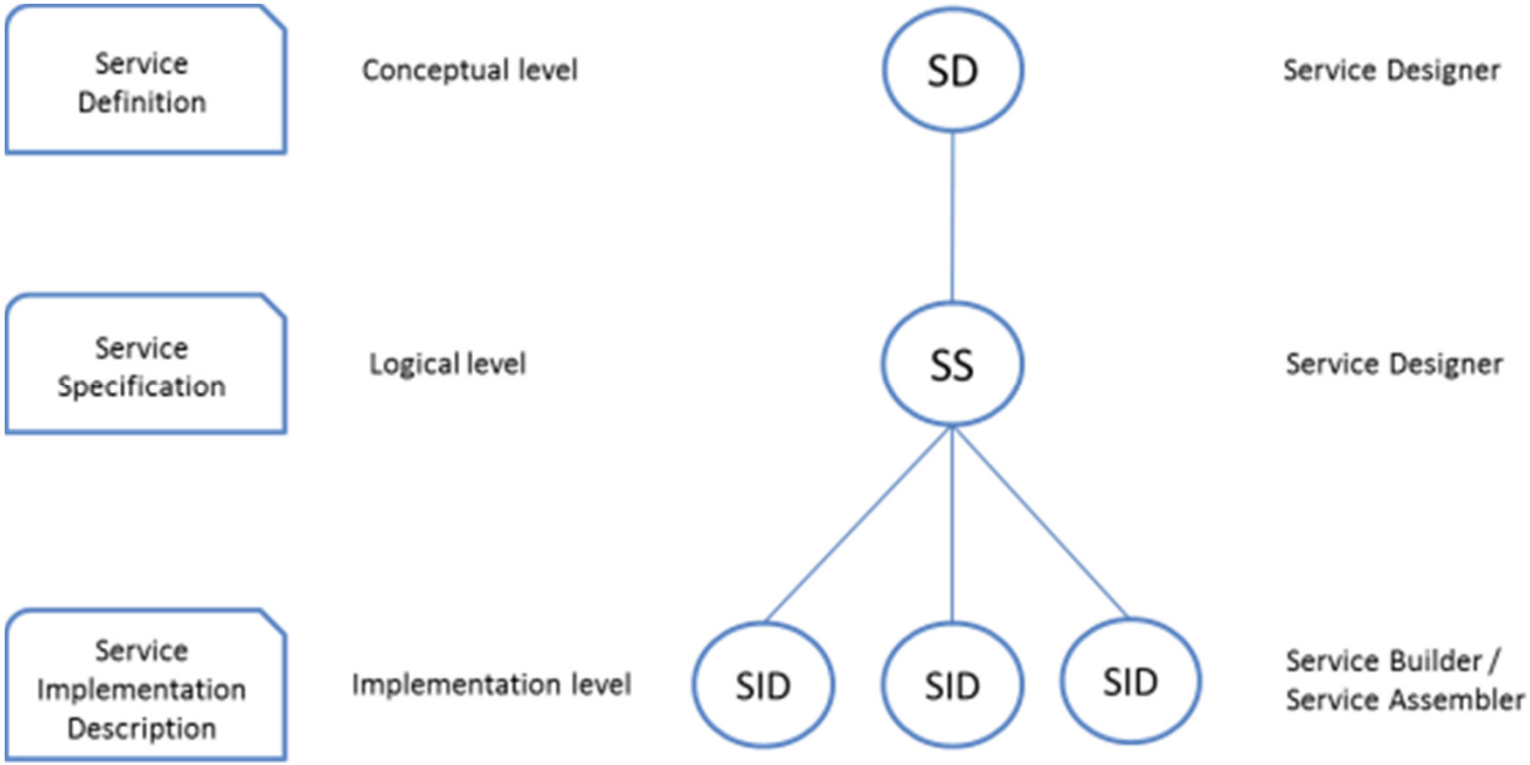
CSPA Proof Of Concept

- Given the short timeframe in which to complete the Proof of Concept, it was decided that the Statistical Services for the Proof of Concept could not be built from scratch.
- Instead, the organisations involved in the project were consulted to find suitable candidate tools/applications that could be wrapped and exposed as Statistical Services.

CSPA PoC: the Tools

- Blaise
  - A data collection, data editing and data processing tool developed by Statistics Netherlands. For the Proof of Concept only the collection function was involved.
- Editrules
  - An error localization tool developed by a staff member at Statistics Netherlands and made available under GPL and can be obtained through the CRAN website.
- CANCEIS (CANadian Census Edit and Imputation System)
  - An editing tool used for error localization and imputation developed by Statistics Canada.
- GCode
  - A generalized automated and assisted coding tool developed by Statistics Canada.
- Statistical Coding Service
  - A coding tool developed by Statistics New Zealand.

Designer


Service Definition
Conceptual level
SD
Service Designer
Service Specification
Logical level
SS
Service Designer
Service Implementation Description
Implementation level
SID
SID
SID
Service Builder / Service Assembler

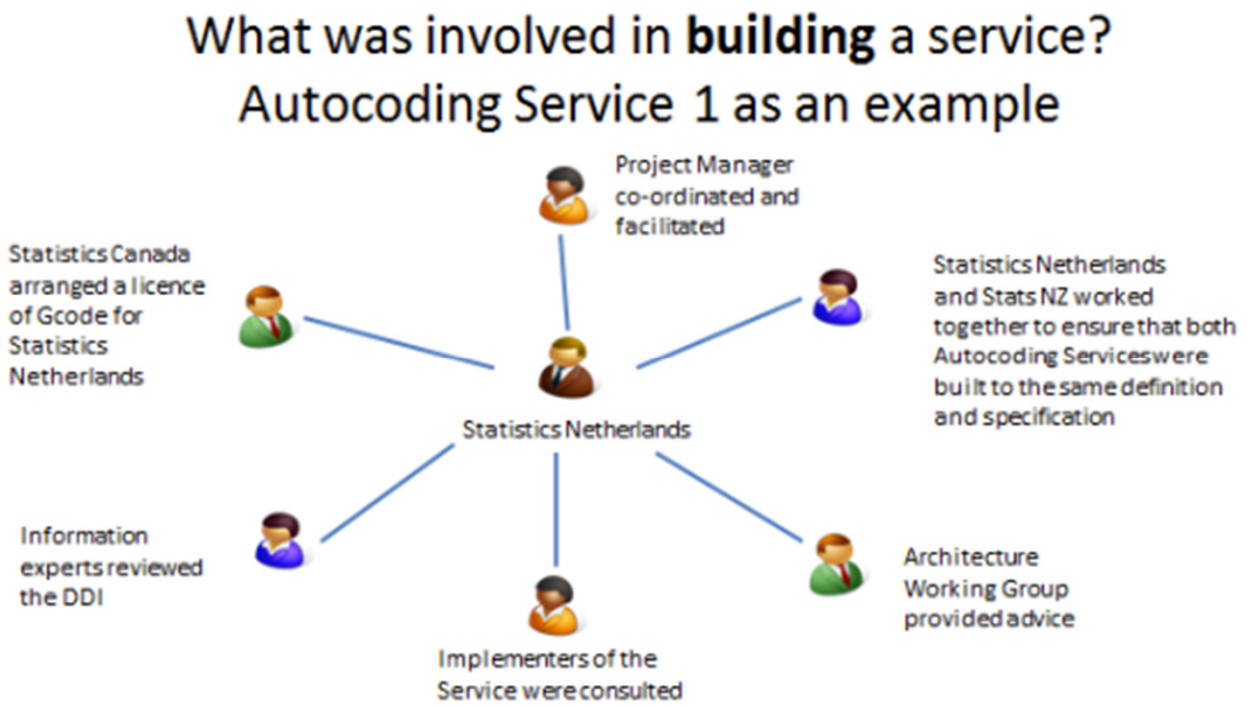
Designer - 2

- The five tools which were to be wrapped for the Proof of Concept performed four business functions:
  - **Run Collection (Blaise)**
  - **Error Localization (Edit Rules)**
  - **Editing and Imputation (CANCEIS)**
  - **Autocoding (GCode and Statistical Classification Service)**
- GCode and the Statistical Classification Service performed the same business function – so at the conceptual and logical level they are the same service.

Builders

- Organizations involved in the wrapping of one of the candidate tools performed the role of "Service Builders".
- Five statistical organizations performed this role during the Proof of Concept
  - **Australia: Run Collection Statistical Service (Blaise)**
  - **Italy: Error Localization Statistical Service (EditRules)**
  - **Canada: Editing and Imputation Statistics Service (CANCEIS)**
  - **Netherlands: Autocoding Service 1 (GCode)**
  - **New Zealand: Autocoding Service 2 (Statistical Classification Service)**

Building a Service: Autocoding 1

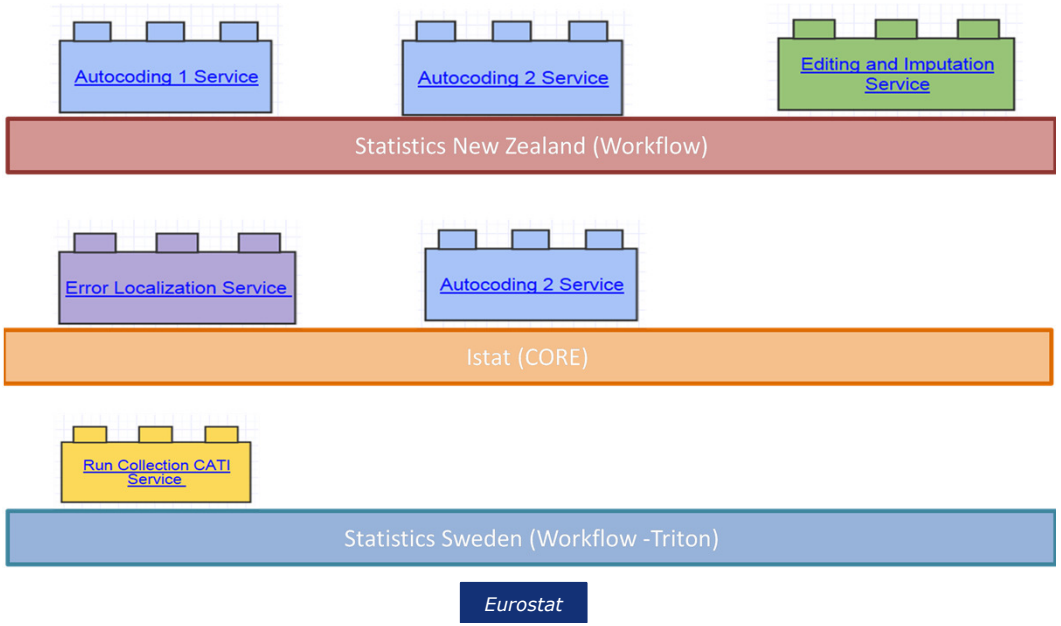


Assemblers

- Within each statistical organization, there needs to be an infrastructural environment in which the generic services can be combined and configured to run as element of organization specific processes.
  - **This environment is not part of CSPA.**
- CSPA assumes that each statistical organization has such an environment and makes statements about the characteristics and capabilities that such a platform must have in order to be able to accept and run Statistical Services that comply with CSPA

Assemblers - 2

- The Statistical Services were implemented (in various combinations, as shown in Figure 5) into three statistical organizations (Italy, New Zealand and Sweden).
- These organizations performed the role of Service Assembler for the Proof of Concept.



Statistical Service Definition Example

Name	Auto coding
Level	Definition
GSBPM	5.2 Classify & Code
Business Function	This Statistical Service maps a field to classification code
Outcomes	This results in a transformed data set that is coherent with the target classification scheme
Restrictions	None
GSIM Inputs	Unit data set, unit data structure, processing activity, classifications, codelist, Rules
GSIM Outputs	Unit data set, unit data structure, number of failed (uncoded) fields
Service dependencies	

Statistical Service Specification

Protocol for invoking the service

- This service is invoked by calling a function called "CodeDataset". There are the following seven parameters (all of them are expressed as URI's, i.e. all data is passed by reference)
  1) **Location of the codelist;**
  2) **Location of the input dataset;**
  3) **Location of the structure file describing the input dataset**
  4) **Location of the mapping file describing which variables in the input dataset to be used**
  5) **Location of the output dataset generated by the service**
  6) **Location of the structure file describing the output dataset generated by the service**
  7) **Location of the process metrics file generated by the service.**
- All parameters are required.
- The protocol used to invoke this function is SOAP, and is in compliance with the guidance provided for developing Statistical Service by CSPA.

Statistical Service Specification

Input messages

- The first four parameters for the service refer to input files. In GSIM terms, the inputs to this service are:

1) a NodeSet consisting of Nodes, which bring together CategoryItems, CodeItems, and other Designations (synonyms).

2) a Unit data set – the texts to be coded for a particular variable

3) a Data structure, describing the structure of the Unit data set

4) a set of Rules, describing which variables the service should use for which purpose.

Statistical Service Specification

Input messages

- The codelist to be passed in must be expressed as a DDI 3.1 instance, using the following structure. The table shows the mapping of the conceptual GSIM objects to their encoding in DDI 3.1

DDI 3.1 Element	GSIM Object
DDIInstance (@id, @agency, @version)	Processing Activity
ResourcePackage (@id, @agency, @version)	[No conceptual object]
Purpose (@id)	[No conceptual object]
Logical Product (@id, @agency, @version)	[No conceptual object]
CategoryScheme (@id, @agency, @version)	CategorySet
Category (@id, @version)	CategoryItem
CategoryName	CategoryItem/Name
Label	Designation
CodeScheme (@id, @agency, @version)	CodeSet
Code	CodeItem
CategoryReference	[Correspondence with CategoryItem in GSIM]
Scheme	[Implementation Specific]
IdentifyingAgency	[Implementation Specific]
ID	[Implementation Specific]
Version	[Implementation Specific]
Value	CodeValue

Statistical Service Specification

Input messages

- The unit data set is a fixed-width ASCII file containing at least a case ID (50 characters maximum) and a variable containing text strings to be coded. Each entry should be on a single line. The corresponding GSIM objects:

Data File	GSIM Object
Unit data set	Unit Data Set
Case ID	Unit Identifier Component
Text string	Attribute Component

Statistical Service Specification

Input messages

- The structure of the unit data set must be expressed as a DDI 3.1 instance, using the following structure. The table below shows the mapping of the conceptual GSIM objects to their encoding in DDI 3.1:

DDI 3.1 Element	GSIM Object
DDIInstance (@ id, @ agency, @version)	Processing Activity
ResourcePackage (@id, @agency, @version)	[No conceptual object]
Purpose (@ id)	[No conceptual object]
Logical Product (@id, @agency, @version)	[No conceptual object]
DataRelationship (@id, @version)	Record Relationship
LogicalRecord (@allvariablesInLogicalRecord="true")	Logical Record
VariableScheme (@id, @agency, @version)	[No conceptual object]
Variable (@id, @version)	Represented Variable/Instance Variable
VariableName	Name
Representation	Value domain
TextRepresentation (@ maxLength)	[No conceptual object]

Statistical Service Specification

Output messages

- The output of the service contains of three files. In GSIM terms, the outputs of this service are:

  5) a Unit data set containing the coded data for the variable concerned;

  6) a Data structure, describing the structure of this Unit data set

  7) a Process Metric, containing information about the execution of the service.

- These generated files will be placed at the locations indicated by the 5th, 6th and 7th input parameters. No return parameter will be generated by the service.

Statistical Service Specification

Output messages

- The unit data set will be a fixed-width ASCII file containing (for the successfully coded entries) the case ID (50 characters maximum) followed by the Code. Each entry should be on a single line.

Data File	GSIM Object
Unit data set	Unit Data Structure
Case ID	Unit Identifier Component
Code	CodeValue

Statistical Service Specification

Output messages

- The structure of the unit data set will be expressed as a DDI 3.1 instance, using the following structure. The table below shows the mapping of the conceptual GSIM objects to their encoding in DDI 3.1:

DDI 3.1 Element		GSIM Object
DDIInstance (@ id, @ agency, @version)		Processing Activity
ResourcePackage (@id, @agency, @version)		[No conceptual object]
Purpose (@ id)		[No conceptual object]
Logical Product (@id, @agency, @version)		[No conceptual object]
DataRelationship (@id, @version)		Record Relationship
LogicalRecord (@allvariablesInLogicalRecord="true")		Logical Record
VariableScheme (@id, @agency, @version)		[No conceptual object]
Variable (@id, @version)		Represented Variable/Instance Variable
VariableName		Name
Representation		Value domain
TextRepresentation (@ maxLength)		[No conceptual object]

Statistical Service Specification

Output messages

- The Process metrics will be expressed as an XML file structured in the following way:

XML Element	Description
CodingMetrics	Container for the coding metrics
Result (@Datetime)	Contains the results of the service execution started at the given date/time
TotalRows	The number of rows found in the input dataset
TotalCoded	The number of successfully coded records

Statistical Service Specification

Error messages

- When the coding process cannot be executed or is aborted due to some error, the service will return an error message. The following error messages can be generated by the service.

Error message	Description
Error in input codelist	The input codelist cannot be read, is syntactically invalid or its content is inconsistent
Error in input dataset	Either, the input dataset, the structure file describing the dataset or the input mapping file cannot be read or contains some error.
Other/unspecified error	Some error occurred during the coding process

Proof of Concept Videos

- Statistics New Zealand
- Statistics Sweden

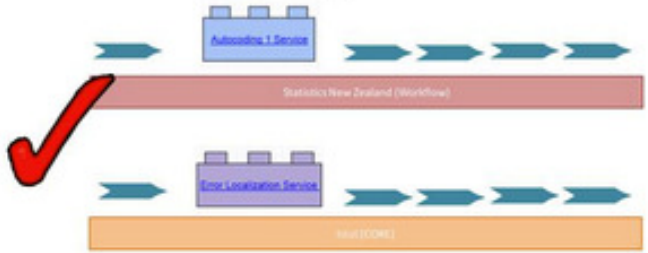
What was proved?

- CSPA is practical and can be implemented by various agencies in a consistent way
- Having tested the architecture, some of the real issues are now known and there is a tested foundation to move forward from.
- One quote from a business perspective on the Proof of Concept was:
  - **"The proof-of-concept form of working with these concepts is in itself very interesting. We can quickly gain insight to both problems and possibilities"**

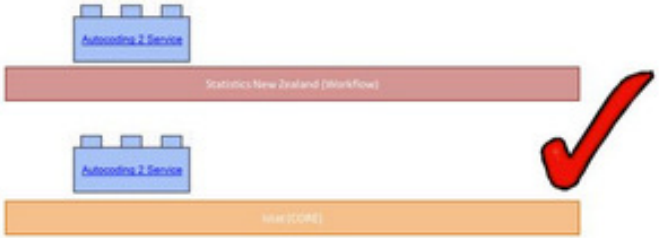
European Commission

What Was Proved

You can fit CSPA Statistical Services into existing processes



CSPA does not prescribe the technology platform an agency requires



You can swap out CSPA compliant services easily

Reusing the same statistical service by configuration

Lessons Learned

- International collaboration is a trade to be mastered
- The on-going contact with colleagues over the globe is stimulating and broadens the understanding. The discussion forum on the CSPA wiki was useful for discussing and progressing issues.
- However, the ability to undertake trouble shooting through the installation / configuration period was made difficult by the time zone differences. It meant that simple problems often took a number of days to resolve.

Lessons Learned - 2

- The separation of roles in design, build, assemble functions worked very well.
- However, due to the limited time spent focused on the Design role (limited to the 1 week design sprint), there was a blurring of the Designer and Builder roles.
- The Service Builders found in some cases that they had to tighten up the design specifications that they were given in order to complete the build work.

Lessons Learned - 3

- Each of the Service Builders and Service Assemblers needed licences for the tools that were wrapped. This was both a challenge and an opportunity.
- Obtaining the licences took some time and caused (small) delays in starting work. This was not a big problem given the small scale of the Proof of Concept.
- However, in the future, if an organisation that owns a Statistical Service has to provide licences for every party who wants to try the service, this could be become onerous.
- Some organisations had processes in place to provide licences and some did not. At least one organisation created a process that they will be able to use for future collaborations.

Lessons Learned - 4

- The Proof of Concept chose to wrap existing tools into Statistical Services for pragmatic reasons. The wrapping did introduce some complexity.
- In some cases, the tools being wrapped by Service Builders were not developed by the organisation performing the role of Service Builder.
- Building service wrappers with meaningful interfaces requires in depth knowledge of the tool being wrapped.
- The Service Assemblers also needed in depth knowledge of Service that they were implementing. Support is required to implement a Statistical Service built by another organization.

Lessons Learned - 5

- The Proof of Concept was one of the first real world uses of GSIM Implementation.
- Support was provided to Service Builders by DDI experts as well as participants in the HLG Frameworks and Standards for Statistical Modernisation project.
- However, Service Builders needed knowledge of GSIM and GSIM Implementation standards (DDI in the case of the Proof of Concept).
- In some cases, DDI needed to be extended. It took time explore the how these extensions should be done.