An OWL Ontology for the Common Statistical Production Architecture

Antoine Dreyer, Franck Cotton, and Guillaume Duffès

INSEE[**], Paris, France
antoine.dreyer@insee.fr, franck.cotton@insee.fr, guillaume.duffes@insee.fr

Abstract. The Common Statistical Production Architecture (CSPA) is a reference architecture for the statistical industry. In particular, CSPA aims at documenting statistical services in a standard way, in order to ease their exchange and reuse between statistical institutes. This paper describes a suggested formalization of some parts of the CSPA specification using OWL. This work led to propose some adjustments that could improve the consistency and clarity of the specification.

Keywords: CSPA, OWL, ontology, statistical service, official statistics

1 Introduction

1.1 The Common Statistical Production Architecture (CSPA)

Since 2010, the United Nations Economic Commission for Europe (UNECE) leads an global effort to develop common standards and models and to foster cooperation in the official statistics community. The flagship of this effort is the Common Statistical Production Architecture (CSPA) [1], a reference architecture for the statistical industry aiming at promoting collaboration between national and international organizations.

CSPA comprises in particular: conceptual data and process models[1], a logical information model (LIM), standard templates for documenting statistical services, standard roles for conceiving, developing and implementing those services, architectural principles at the application and technology levels, etc.

This different components do not exhibit the same level of formalization. The information models are expressed as UML, and work has already been conducted to represent the generic process model using OWL [2]; on the other hand, the templates for documenting the statistical services are just text documents, and it is obvious from the existing examples of CSPA services that they were and still are interpreted in various ways. Existing service documentation show also that there are different views on the expected granularity of CSPA services.

To address these problems, a more formal representation of some parts of the CSPA specification is needed. This is what we are aiming at in this paper.

[**] Institut National de la Statistique et des Études Économiques, `http://insee.fr`
[1] Resp. GSIM (`http://www1.unece.org/stat/platform/display/gsim/Generic+Statistical+Information+Model`) and GSBPM (`http://www1.unece.org/stat/platform/display/metis/The+Generic+Statistical+Business+Process+Model`).

1.2 What do we want to do?

We want to formalize the CSPA specification using OWL [3]. OWL is based on RDF and "*is designed to represent rich and complex knowledge about things, groups of things, and relations between things*". Several reasons led us to select OWL rather than other modeling formalisms:

- As a formal approach of knowledge, OWL brings structure and consistency to prior knowledge. This helps avoiding misunderstandings, brings uniformity and facilitates the exchange of information, which is at the core of CSPA.
- RDF is machine-actionable, so it can be used directly to capture, exchange or disseminate common information and to build information systems.
- The main models defined in CSPA (GSBPM and GSIM) are now expressed in OWL: using the same formalism is clearly a factor of interoperability.

OWL has well-known shortcomings when it comes to data validation[2], but validation is not an important objective at this stage. We are starting from a material which is not structured at all, so the first goals are to ease sharing of semantics, and to progress towards a uniform representation of CSPA.

We should note that applying formalism on existing material can raise difficulties. Sometimes, adaptations to the CSPA concepts or vocabularies would ease the process. Even if our main goal is not change to CSPA specification but to foster its use within the statistical community, we had to suggest some changes and leave open some questions. This will be reported back to the CSPA management team.

1.3 What is CSPA for?

The goal of CSPA is not unique. On the one hand it is a representation of processes, on the other hand it is a way to share IT components between statistical institutes. This duality obliges us to be cautious: if the ontology is too conceptual, few people will use it; if it is not conceptual enough, it will not allow efficient exchange of information. In order to evaluate the right level of representation, we first studied the CSPA descriptions already provided by several institutions [4]. We observed that the original CSPA specification was not always tailored to every need, and consequently chose to take a user-driven perspective in order to provide a way to specify as close as possible to the existing examples. This led us to add some aspects to the original CSPA specification, in particular the distinction between functions and packages, which is explained below.

2 Structure of CSPA

2.1 Introduction

We decribe in this section the different notions defined by the CSPA specification, in order to prepare the development of the ontology, which will be described in the next section.

[2] See for example `https://www.w3.org/2012/12/rdf-val/`

2.2 Three levels of description

CSPA distinguishes between three levels of service description, and associates predefined organisation roles to them:

Service definition (SD) is the conceptual level which is human-oriented and has less details. It is the user's view, independent of physical implementation. It is made by a Service Designer, using conceptual models like the GSIM [5] as information model, and like GSBPM [6] for processes.

Service specification (SS) is the logical level. It ensures that the service description meets the requirements of CSPA. It is made by the Service Designer, using logical models like the CSPA LIM or the information models associated with DDI[3] or SDMX[4].

Service implementation description (SID) is the physical level which is computer oriented and has most details. It is made by a Service Builder or Service Assembler, using (among others) DDI or SDMX instances.

2.3 Distinction between functions and packages

We can see from existing examples that current implementers of CSPA want to specify two slightly different things, namely packages and functions. One service may actually be a bundle of functions, potentially accessible via different protocols. Thus we need to describe not only the bundle of functions as one entity but also each and every function in the bundle. Depending on the context, the bundle can also be seen as an independant software or only a package. To help our analysis, we studied different package description systems, in particular: JavaDoc[5], Debian Control File[6] (also used by R) and Python[7]. As a result, we introduced a distinction between functions and packages, even if the current CSPA specification does not make this distinction clearly. We split the 3 levels of CSPA into 2 parts: one for functions and one for packages, as is explained further below.

2.4 Mixing functions and packages with the levels of description

We first look at the original CSPA structure with 3 levels (cf Fig. 1), where one service definition points to one service specification which points to one or more service implementation descriptions.

Then we introduce the distinction between function and package. In Fig. 2, the 3 levels of description are encapsulated inside package or function containers; one package container points to several function containers.

[3] Cf. `http://ddialliance.org`
[4] Cf. `http://sdmx.org`
[5] Cf. `http://www.oracle.com/technetwork/articles/java/index-137868.html`
[6] Cf. `https://www.debian.org/doc/debian-policy/ch-controlfields.html`
[7] Cf. `https://pypi.python.org/pypi?%3Aaction=list_classifiers`

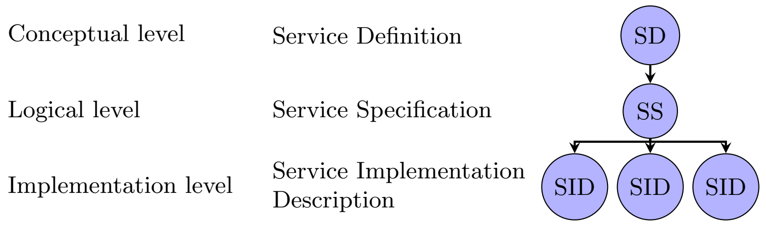

Fig. 1. Minimal Linkage between levels of description

This is a very theoretic representation, because in a typical situation, not every level exist for each function or package. Those missing levels (service specification for the CSPA package, for instance) lead to a difficulty: the initial schema imposes sequential links (service definition linked to service specification linked to service implementation description). That is why we must adopt a slightly different structure (which is exemplified in Fig. 3), where each level inside a container is linked to the container itself and not to another level.

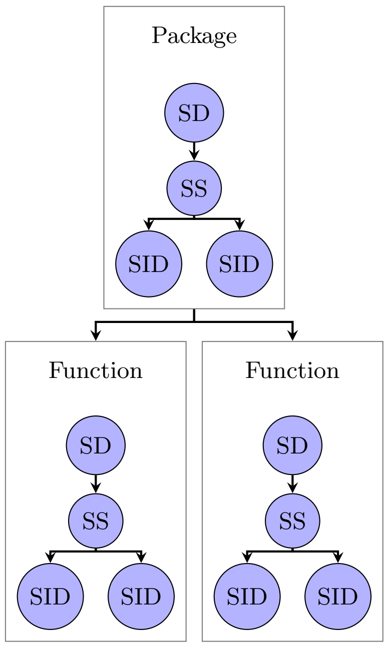

Fig. 2. Theoretic linkage between packages, functions and levels of description

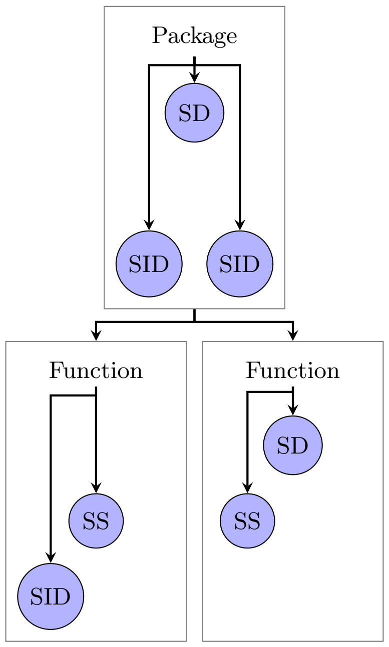

Fig. 3. Example of linkage between packages, functions and levels of description

2.5 Properties and topics

The CSPA specification defines different properties for the services, many unique to one level of description. To ease understanding, we group those properties into 8 different topics:

Business function (only at conceptual level, mostly for packages) to describe what the service is for.
Documentation to document the service (how to use it for instance).
Provenance to describe where the service and its description come from.
Interface (mainly for packages) to describe how to invoke the service, for instance using Bash or a REST API
Dependencies (mainly for packages) to detail what is needed to invoke the service, for instance a specific operating system or database; this can be linked to an Interface at implementation level.
Input (only for functions) to describe the inputs of a function.
Output (only for functions) to describe the outputs of a function.
Identification (Name and Version), to identify a service unambiguously.

We list below the initial properties contained in the CSPA specification, organized according to the topics that we introduced previously. At the beginning of the line is the name of the properties topic described earlier; the names of the initial properties follow. The full initial templates of CSPA with definitions are available on the UNECE website [7].

Service definition initial properties

Identification Name
Business function Business Function, GSBPM, Outcomes, Restrictions
Dependency Service dependencies
Input GSIM Input
Output GSIM Output

Service specification initial properties

Identification Name
Interface Protocol for Invoking the Service
Input Input Message
Output Output Message
Documentation Applicable Methodologies

Service implementation description

Identification Name, Version
Interface Invocation protocols, Service Interface
Input Data-by-Reference protocols
Documentation Additional information, Installation documentation
Provenance Builder Organization
Dependency Technical dependencies

Most of those topics are often present in each of the 3 levels (definition, specification, implementation).

2.6 Roles

As already mentioned, CSPA defines [8] organizational roles that are associated with the life-cycle of statistical services. Those eight roles are:

- Assembler,
- Builder,
- Configurer,
- Designer,
- Environment Provider,
- Investor,
- Service Provider,
- User.

3 Ontology description

3.1 Introduction

At this stage, we have distinguished in CSPA three main semantic axes related to the statistical services: level of description, service granularity and property topics. Consequently, we will organize the global structure of the ontology with 3 levels of description (service definition, service specification and service implementation description), 2 granularity categories (functions and packages), and 8 topics (Business function, Documentation, Provenance, Interface, Dependency, Input, Ouput, Identification).

We have also identified additional concepts present in the specification, like organizational roles.

In this section, we decribe how these different notions are formalized in the CSPA ontology that we propose. Due to space constraints, we limit ourselves to a narrative description: the actual code of the ontology, as well as an illustrative example on one of the existing CSPA services, can be found on GitHub[8].

3.2 Classes and properties for levels and categories

In order to describe granularity categories, we create two classes: Package and Function, and they are defined as subclasses of a common PackageOrFunction class. Similarly, ServiceDefinition, ServiceSpecification and ServiceImplementationDescription classes are used to represent the levels of service documentation, and they are made subclasses of a common DescriptionLevel class. Since some properties are only found for one level and one category, we need to create 6 (2x3) classes mixing categories and levels. We also need 6 corresponding properties to link each category to its 3 description levels. Finally, we need a property to link a function (or a package) to a package: a package can contain functions or other packages.

[8] Cf. `https://github.com/FranckCo/Stamina/blob/master/doc/cspa`

3.3 Properties and topics implementation with OWL

In an OWL ontology, properties define how information is stored (DataType properties) or how objects (instances of classes) are linked together (Object properties). Consequently, all CSPA properties are implemented using OWL DataType properties or Object properties.

How we can represent topics is not as straightforward. All topics are groups of properties bringing knowledge on the same theme, but some are more than that: they also represent objects. For instance, the Input topic represent what a function receives to operate, but a function also has input parameters which are objects on their own. It is thus logical to merge the two notions: a function will be linked to one or more objects, instances of an Input class.

On the whole, four topics (Interface, Dependencies, Input, Output) can be seen as objects: an Interface is one of potentially several ways to invoke the service, a Dependency is one thing out of many needed to use the service, an Input is an object consumed by a function, an Output is an object produced by a function). OWL classes are needed for representing each of those 4 topics.

For the other topics (Business function, Documentation, Provenance, Identification), there is no real need of OWL classes; properties could be linked directly to the function or the package class. But since there are many properties, we think that it is clearer to create classes also for these topics and link properties of a topic to their topic class, that acts as an intermediary for organizing the information. This structure intends to inform the user where to look at for a specific information, and thus to facilitate the understanding and use of the ontology. We think the added clarity is worth the small extra complexity in the structure.

We make an exception for Identification: since identification should be fast and straightforward, properties in the identification topic should be accessible directly, and it is better to attach them directly than through a topic class.

3.4 Classes for topics and levels

According to the conclusions of the previous section, we created in the ontology a PropertyTopic class to contain property topics, and we created 7 subclasses of the PropertyTopic class, for each of the topics except Identification. The identification properties are described below.

We must take into account the interaction between property topics and levels. The BusinessFunction topic can be applied only to one level: ServiceDefinition. Provenance and Documentation can be applied to all levels, and no distinction is made between levels in regard of the content of those topics.

Dependency, Interface, Input and Output topics, on the other hand, can be applied to all levels, but properties relative to each level are different: the content of these topics is not the same in the different levels, so we have to create 12 classes combining topics and levels (4 topics and 3 levels). Each class is a subclass of the topic it refers to, and its name is the concatenation of the level name (abbreviated as Definition, Specification and Implementation respectively) and of the topic name. For example, the DefinitionInput class is a subclass of the Input class.

As of today, the subclasses for each level for Dependency and Interface are exactly the same, but the CSPA specification shows that it may not stay true in the future. Table 1 illustrates the connections between topics and levels of description in CSPA.

With these different definitions, we can now link one level, or all the levels, or the levels concerning only functions, etc., to properties topic. Also, to facilitate the future use of the ontology, we add a description property to every property topic, which allows to freely add descriptive text to the topic.

Table 1. Properties linking levels and topics: levels are by row, topics are by column, a blue cell indicate the existence of one or more property.

	Provenance	Documentation	BusinessFunction	Dependency	Interface	Input	Output
DescriptionLevel	🟦	🟦					
ServiceDefinition			🟦	🟦	🟦		
FunctionDefinition						🟦	🟦
ServiceSpecification				🟦	🟦		
FunctionSpecification						🟦	🟦
ServiceImplementationDescription				🟦	🟦		
FunctionImplementation						🟦	🟦

In table 1, the lines are the OWL classes corresponding to level of description and glanularity categories. The blue cells only indicate the highest-level class to which a given topic is attached. For example, Provenance and Documentation are attached to DescriptionLevel, which is a common ancestor of all the others, and so these topics are defined for all level and all categories. FunctionDefinition inherits the topics from DescriptionLevel and ServiceDefinition, and adds Input and Output. PackageDefinition is not represented because it has only inherited topics.

3.5 Topic identification and generalization

For the 4 topics referring naturally to objects that we listed before (Interface, Dependency, Input and Output), one instance can be described in more than one level. An obvious need, and thus ontology use case, is to be able to capture the relation between the corresponding objects at different levels. For instance, it should be easy to link with each other the different levels of description of an Input (e.g. a GSIM object at the definition level and a LIM object at the specification level).

Different possibilities were considered to implement this feature, and we settled on the use of a common identifier. In order to give flexibility at implementation time, we modeled the identifier as an OWL class, with subclasses for each topic (InputIdentifier, etc.). Only a label property is defined for the identifiers, but other properties can be added in the future for more specific identification.

Regarding the ranges of the topic classes, table 1 shows that we could have specified them very precisely. Nevertheless, we felt that it was clearer to provide topics as close as possible for each level, so that we decided to extend the range of some topics. Only business function topic is not extended to every level, since we considered that the business function was only conceptual and did not have an implementation. The goal of extending topics is not to alter CSPA specification, but to make it clearer (every topic present at each level), and easier to use.

3.6 Identification properties

As we mentioned above, the Identification topic has no associated topic class: in order to make theme as accessible as possible, the identification properties, label and version, are directly linked to the objects they qualifiy. They are defined as datatype properties with range xsd:string and can be applied on DescriptionLevel and PackageOrFunction (and thus all their subclasses). A label can also be applied on Identifier, Dependencies, Interface, Input, Output classes. As a consequence, for Dependencies, Interface, Input, Output classes, a label can be given directly or via an Identifier.

3.7 Additional components and features of the ontology

We decribe in this section more classes and properties dedicated to modeling the CSPA roles, as well as the specific properties corresponding to the topics introduced in 2.5.

Roles and provenance We introduced the CSPA roles in sub-section 2.6. To represent these roles and their action in the life-cycle of a statistical service, we use the PROV[9], ORG[10] and FOAF[11] ontologies. The whole ontologies could be used to document provenance information, but only a few properties and classes correspond to the CSPA initial specification.

In the rest of the paper, the usual prefixes prov:, org: and foaf: will be used for the corresponding ontologies. The cspa: prefix will be used for the CSPA ontology.

In order to incorporate PROV, we define a cspa:Provenance class (the name will be precised in a future version) as a subclass of the prov:Entity class. The property prov:wasGeneratedBy allows to document an organization as a

[9] Cf. `https://www.w3.org/TR/prov-o/`
[10] Cf. `https://www.w3.org/TR/vocab-org/`
[11] Cf. `http://xmlns.com/foaf/spec/`

CSPA role. We also create a cspa:Organization sub-class of prov:Organization and org:Organization (this class being itself a sub-class of foaf:Agent).

We also create a cspa:organizationName property, which is an alias of foaf:name.

Finally, we introduce 8 properties linking cspa:Provenance to each role, all sub-properties of prov:wasGeneratedBy.

For instance, one may describe a CSPA service documentation made by a National statistical institute (NSI) in the following way (expressed in the Turtle language[9]):

```
:something a cspa:DescriptionLevel;
cspa:comesFrom [ a cspa:Provenance;
cspa:builderOrganization [ a cspa:Organization;
cspa:organizationName "NSI" ] ] .
```

Documentation The ontology defines 5 properties for this topic:

- cspa:installationGuide, which links to a foaf:Document
- foaf:homepage, which links to a foaf:Document
- cspa:methods, with string values
- cspa:description, with string values
- cspa:additionalInformation, with string values

BusinessFunction We define 3 specific properties for this topic. cspa:outcomes and cspa:restrictions are datatype properties with string values. The cspa:gsbpmSubProcess property links to the SubProcess class in GSBPM ontology [2].

It should be noted that the business function topic in CSPA is not exactly the same as in GSIM [10], but we underline the proximity of the two concepts by making the cspa:BusinessFunction class a subclass of a GSIM business function class that will be defined in a future work on a GSIM ontology.

Dependency and Interface Since the CSPA specification is not very precise on the information that should be stored in these topics, and because we do not want to have too many properties, we chose in a first approach to have a very simple structure, where a dependency or an interface is described with only two properties: a label and a description, both string datatype properties. In consequence, there is no property specific to Dependency and Interface, except for identifiers (see above).

We also introduce a direct link between Dependency and Interface. At definition and specification levels, dependencies should be broad enough not to depend on a specific interface. At implementation level however, a direct link between dependency and interface is needed. For instance, if the service has several interfaces including a web interface, then a dependency concerning compatible browser only makes sense for the web interface.

As a consequence, we extend the cspa:implementationDependsOn property that was previously created to represent the a service implementation and a dependency, so that its domain also covers the cspa:ImplementationInterface class.

Input and Output It would be possible to use for the Input and Output topics the same label/description construct that we used for Interface and Dependency, but being able to connect services together is a key point of CSPA, so that is not a recommended method.

We saw that service definition level, Input and Output instances should be GSIM objects. Thus, we introduce a generic gsim:gsimObject class and create two object properties, cspa:gsimInput and cspa:gsimOutput with this range, defined on the cspa:DefinitionInput and cspa:DefinitionOutput topic classes.

A similar construct is used for the specification level with CSPA LIM objects.

On a service implementation level, the way specification should be done is open: for example, is possible to use GSIM Process Input or a GSIM Process Output objects. At this level, it is possible to link an input or an output directly to an Interface; some parameters may indeed only have sense for a specific interface.

Cardinality restrictions More experience with CSPA is needed in order to define cardinality restrictions in the ontology, and we chose not to use owl:Restriction to allow for any future implementation of them. However using a property more than once would never produce non-sensical specification.

4 Conclusion ad future work

We presented in this paper the construction of an OWL ontology aiming to model the core of the Common Statistical Production Architecture specification, namely the different levels of service documentation and the organizational roles involved in the conception, building and implementation of statistical services. This work consisted in a detailed semantic analysis of CSPA, allowing to identify the main concepts in order to translate them as OWL classes and properties. We had to make some compromises and some additions to CSPA in order to build a complete and consistent ontology, but we are confident that this will enhance the coherence between CSPA and its main pillars: GSBPM and GSIM.

The work presented here is just a first step; it will be prolonged in different directions.

First, since more CSPA services are presently being developed by different organizations, we will shortly be able to test our model on more examples. Currently, there are only a handful of service descriptions, so a serious evaluation of the model is not possible, although we found it performed well on existing cases (see example referenced above). With more service descriptions, we will be able to evaluate our work more completely, which is likely to help us refine the ontology.

Second, the results of our work will be presented to the CSPA sponsors, specifically the CSPA Implementation Group which is the high-level committee in charge of maintaining CSPA. We believe that this input will be taken into consideration for a future version of CSPA.

Third, the current work is part of a broader interest in semantic business process management (SBPM). The article by Hepp and others [11] points to the leverage provided by a semantic approach of business process management. The article by Bevilacqua and others [12] show an example of service composition using service description made with OWL. Some additional work is needed in order to incorporate in the CSPA ontology the outcomes of these works.

Finally, we intend to operationalize the ontology by constructing an IT system based on it. A hackathon session sponsored by the UNECE is already scheduled with colleagues from different countries, and one objective is to develop a graphical CSPA service editor that will produce representations conformant to the ontology presented here.

References

1. *Common Statistical Production Architecture*, `http://www1.unece.org/stat/platform/display/CSPA/Common+Statistical+Production+Architecture`
2. Cotton, Franck and Gillman, Daniel, *Modeling the Statistical Process with Linked Metadata*. In: Proceedings of the 3rd International Workshop on Semantic Statistics, `http://ceur-ws.org/Vol-1551/article-06.pdf`.
3. *W3C Web Ontology Language*, `https://www.w3.org/OWL/`
4. *Implementing CSPA Projects*, `http://www1.unece.org/stat/platform/display/CSPA/Implementing+CSPA+Projects`
5. *Generic Statistical Information Model*, `http://www1.unece.org/stat/platform/display/metis/Generic+Statistical+Information+Model`
6. *The Generic Statistical Business Process Model*, `http://www1.unece.org/stat/platform/display/metis/The+Generic+Statistical+Business+Process+Model`
7. *CSPA, Annex 1: Templates*, `http://www1.unece.org/stat/platform/display/CSPA/Annex+1_+Templates`
8. *Roles involved in the production of IT enabled CSPA Services*, `http://www1.unece.org/stat/platform/display/CSPA/Roles+involved+in+the+production+of+IT+enabled+CSPA+Services`
9. *Turtle (Terse RDF Triple Language)*, `https://www.w3.org/TR/turtle/`
10. *Clickable GSIM: Business Function*, `http://www1.unece.org/stat/platform/display/GSIMclick/Business+Function`
11. Martin Hepp Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensell, *Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management*, `www.heppnetz.de/files/mhepp-et-al-SemanticBusinessProcessManagement.pdf`.
12. L. Bevilacqua and A. Furno and V. S. di Carlo and E. Zimeo, *A tool for automatic generation of WS-BPEL compositions from OWL-S described services*. In: 2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA), `http://angelofurno.net/documents/WS-BPEL_Composition_SKIMA11-v1.4.pdf`.