Automotive ranges as eCommerce data

François-Paul Servant, Edouard Chevalier and François Jurain

Renault / Dir. de l'Informatique / Service Intelligence Artificielle Appliquée
{francois-paul.servant,edouard.chevalier,francois.jurain}@renault.com

Abstract. Exposing data about customizable products is a challenging issue, because of the number of features and options customers can choose from, and because of the intricate constraints between them. The lack of a well-established way to publish such data on the web impedes the development of the e-business-related benefits that we could expect. Building on previous work at Renault, this paper shows that a few additions to Schema.org could foster the publishing of data about complex products such as new cars.

Key words: Structured eCommerce data, Configuration, Automotive, Linked Data, Schema.org, GoodRelations

1 Introduction

The publishing on the web of structured data about products has become mainstream, thanks to the Schema.org initiative and to the GoodRelations vocabulary. It allows better e-business performance by increasing the visibility of commercial offers. This improves in particular the accuracy of search engines: when instructed to index well identified and precisely described products, they can add them to their "knowledge graph" of known entities. This has interesting results for products such as books: search results list actual products rather than mere web pages, and include links to commercial offers with price, ratings, etc.

However, suppose you are looking for, say, a small car with a gasoline engine, a sun-roof and a navigation system, and you are concerned with the price and with CO_2 emissions; your search engine probably won't help you there.

This is hardly surprising: cars are more complicated to describe than books, and the data needed to respond to such requests is simply not online. Books are searched on the basis of a very small set of properties (title, author,...); they are well identified, e.g. through ISBN; and comparisons between commercial offers only involve completely defined products. In contrast, cars are customizable, a crucial aspect of the problem: rather than fully specified products, what you compare are sets of them, that is, partially defined products.

The description of automotive ranges raises some challenging issues. Cars indeed have many optional features, and ranges of new cars are therefore huge: more than 10^{20} different cars are for sale at Renault. Furthermore, and this is the tricky point, there are constraints between the features that invalidate many of their combinations: only one chance in 100,000 to define an existing Renault car, if you pick at random from the available features without taking the constraints into account. The definition of a range of new cars is a “Constraint Satisfaction Problem”: it cannot be represented accurately as a simple tabular data-sheet that just lists some predefined properties of car models. It could be represented by means of Semantic Web languages [3], but only sophisticated automatic reasoning would make these data usable in practice, a capability that cannot be expected from simple agents. This rules out the option, when publishing data on the web.

The fact is, there is no shared, well established way to publish data about cars. Schema.org includes nothing about vehicles, and nothing about the handling of partially defined products in general, a key point when describing ranges of new cars. This impedes the publishing of such data: no how-to on one hand, nothing to expect in terms of SEO on the other hand.

It can be done, nonetheless: to facilitate the sharing of data between in-house systems and web applications, Renault publishes structured data on the web that describe its commercial range[1] (as pure RDF[2], as RDFa markup in HTML[3]). This work contributes a domain-independent solution for the description of ranges of customizable products, based on their modeling as graphs of “Partially Defined Products” (PDP), with each PDP (or Configuration) linking to those that refine it [2]. This is formalized in the “Configuration as Linked Data Ontology” (“COLD”)[4], a lightweight vocabulary (3 core classes, 5 properties) available under a Creative Commons license.

This paper builds on these results to propose a lightweight extension to Schema.org to make it support PDP handling. It is structured as follows: section 2 lists related works. Section 3 summarizes the important points of our earlier paper about the configuration process as linked data [2]. In section 4, we discuss the fact that the solution, being generic, is meant to be complemented by domain-dependent vocabularies. Section 5 takes the point of view of a search engine: the published data being huge, can it be effectively indexed? Finally, section 6 proposes a minimal extension to Schema.org.

2 Related Work

The GoodRelations (GR) ontology has become a de-facto standard for the publishing of e-business data on the web, a status reinforced by its integration into Schema.org. GR allows the description of product offerings [1]. In a clean separation of concerns, it has been designed to be used in combination with additional ontologies for product types and their properties, such as the “Product Types Ontology”[5].

[1] http://purl.org/configurationontology/quickstart
[2] http://uk.co.rplug.renault.com/product/gen?as=ttl
[3] http://uk.co.rplug.renault.com/product/gen?embed=true
[4] http://purl.org/configurationontology
[5] http://www.productontology.org

The main subject of this paper is therefore the question of the use of PDPs in GR descriptions. This problem seems not yet solved, nor substantially addressed, considering that, e.g., the GR cookbook for "product variants"[6] is empty to date. It could be argued that GR's ProductOrServiceModel class, an abstraction used to model "prototypes" of products, along with the ability to derive descriptions of actual products from it, could be used for the purpose of describing PDPs, but it requires non-standard reasoning.

Product ontologies developed as extensions to GR, such as those from the OPDM project[7], or the "Vehicle Sales Ontology"[8] (VSO), seem to define properties only meant to be used in the descriptions of completely defined products; in spite, in the case of VSO, of its focus on a domain where product customization is an important topic.

In the precise context of our work - the description of ranges of new cars - the main contribution that we know of is Volkswagen's "Car Option Ontology"[9], an extension to VSO. The approach is different from the Configuration Ontology we advocate: they include the constraints between options in the publication, using a proprietary vocabulary. This puts the burden of the reasoning on the client, requiring it to have reasoning capabilities.

3 Configuration as Linked Data

3.1 Principles

As ranges of customizable products are too large to be enumerated, they are defined in intention. The description of a family of similar products (typically those of the same "model") is based on a "lexicon", i.e., a set of variables representing the relevant descriptive attributes: body type, type of fuel, color, etc. In a completely defined product, each of these variables is assigned only one value. Such a value is called a "specification" in ISO-10303 STEP AP 214 terminology, a term that we have retained in the definition of our ontology. Then a set of constraints restricts the possible combinations of specifications. The definition of a range of customizable products is therefore a Constraint Satisfaction Problem (CSP) - a class of problems well known to be computationally hard.

The configuration process, which helps a customer to make her choice among a range of customizable products, one step at a time, feature after feature, can be modeled as the traversal of a graph of "Partially Defined Products" (PDP), or "Configurations", each configuration linking to those that refine it. Each Configuration, that is, each step of the configuration process, is identified by the list of the features selected so far. When accessing the corresponding URI, we get the data that describe the Configuration, in particular the links that allow to select among the remaining choices.

[6] http://wiki.goodrelations-vocabulary.org/index.php?title=Cookbook/Variants
[7] http://www.ebusiness-unibw.org/ontologies/opdm/
[8] http://purl.org/vso
[9] http://purl.org/coo

The “Configuration as Linked Data” ontology (COLD) describes the classes and properties involved in the modeling of the configuration process as Linked Data. It is really simple, with three main classes (Configuration, Specification and ConfigurationLink), and a few properties that model the state of a specification with respect to a given configuration: is it chosen? implied? possible? etc.

Let’s take an example to make things clear: a very simple range of cars, consisting of only one model (“foo:Model1”); a customer can choose the fuel type (either diesel or gasoline), and the gearbox type (automatic or manual), with one constraint: the automatic gearbox is only available with a gasoline engine (the total number of different completely defined cars is therefore 3). A configurator application identifies all partially defined configurations, and describes them with COLD. For instance, the “Model1 with a diesel engine” (remember that its gearbox is necessarily a manual one):

```
foo:Model1DieselConf cold:chosenSpec foo:Model1, foo:Diesel ;
    cold:impliedSpec foo:Manual.
```

On the “Model1 with a gasoline engine”, a customer still can choose the gearbox:

```
foo:Model1GasConf cold:chosenSpec foo:Model1, foo:Gasoline;
    cold:possible [a cold:ConfigurationLink;
        cold:specToBeAdded foo:Manual;
        cold:linkedConf foo:Model1GasManualConf];
    cold:possible [a cold:ConfigurationLink;
        cold:specToBeAdded foo:Automatic;
        cold:linkedConf foo:Model1GasAutoConf].
```

We see here the linked data nature of the representation of the range: each value of the “possible” property associates a specification that can be chosen (e.g. foo:Manual) and the link to the corresponding refined configuration (foo:Model1GasManualConf).

3.2 Salient points

- Configurations are first-class objects, identified by URIs, and are therefore easily shared between applications.
- The complexity of the range is hidden from the client. All reasoning takes place inside the service publishing the data, no reasoning capability is required from the client agent.
- Ranges can be crawled, either starting from the root of the dataset or from any configuration, and following links whose semantics is precisely defined.
- The approach is domain-independent.
- It integrates nicely with GoodRelations (in GR terms, a Configuration is a ProductOrServiceModel).

4 Domain-dependent thesauri

The “Configuration as Linked Data” ontology is generic: it is independent of the variables and specifications that define a product. In other words, it provides a framework that needs to be complemented with dedicated, domain-dependent vocabularies, to define the specifications. This pattern should not be a problem, thanks to Schema.org’s “additionalType” property: the Product Types Ontology follows the same pattern vis-a-vis GoodRelations.

Note the shift from vocabularies aimed at describing products, with fine-grained properties such as “fuelType”, to vocabularies aimed at defining instances and classes of specifications, such as “FuelType”. This shift is needed to describe PDP’s, because a PDP is more than a Completely Defined Product with some properties left undocumented: its description requires the ability to state more than one kind of relationship between a PDP and a given Specification. This shift, which appears to be a very natural one anyway, offers great flexibility when describing products, including completely defined ones; e. g. it allows to define hierarchies of classes of Specifications (such as Sunroof>ElectricSunroof).

Let us also note that the COLD framework allows publishers to use their own terms when describing their products. This is an important point, because:

– the whole purpose of the configuration process is to produce an order for a completely defined product; which implies its definition in the manufacturing company’s terms,
– no precision is lost, in contrast to what would happen if we had to map to a different vocabulary (no vendor will downgrade the description of his own product for the sole purpose of making it comparable to others’ products),
– it makes it possible to publish the data as they are in the publisher’s systems, at no additional cost, still leaving open the option to enhance the published data later.

5 Indexing Configurations

COLD gives us the means to precisely describe ranges of customizable products, making it easy to crawl them, following links whose semantics is precisely defined. Our ranges are very large, though; we avoid the complexity of describing and handling a CSP through an increase in the size of the published data: this is the price of simplicity.

Therefore, the indexing of configurations by search engines can only be partial. Does it matter? The typical person searching for a car will enter a limited number of specifications: “a small car with gasoline engine, sunroof and air conditioning”, for instance. Even if they vary from person to person, the set of popular features will probably be small, because not all the specifications are of equal interest: the sunroof, the navigation system, etc. are probably more important - for a customer as well as for a search engine, or a vendor - than, say, the capacity of the fuel tank; usually, people do not search for “negative specifications" (such as "without sunroof"); moreover, the popular features are a function of the car model: when indexing configurations corresponding to low-end models, air conditioning may be important; not so with high-end models, where it is always included. So, when indexing configurations corresponding to a given model, it is probably enough to only index the configurations including up to, say, 4 features chosen among 15; amounting to a quite manageable set of less than 1901 configurations to index, per car model. Thus, for any car model, search engines will be able to store and return the exact configurations matching the conjunctions of popular terms they expect to find in user queries; then, users will be just one click away from the corresponding pages in vendors' configurator web applications, a major improvement in the precision of search results.

So, search engines can make effective use of configuration data, by choosing the links they want to follow in the dataset, and by stopping when they want. Publishers of these data can also limit the links they provide in the data, or give clues to search engines; for instance, with the "no-follow" directive if the data is provided as markup within HTML.

6 Proposal for Schema.org

As COLD is very simple, it could serve as a basis for the description of customizable products in Schema.org, which currently provides no way to handle them. It can be streamlined further: some of the properties are specific to advanced configurator applications and can be left out (e.g. the "impossible" property that allows a user to choose a Specification incompatible with its previous selections, at the cost of discarding some of them). Also, as a Configuration is essentially a ProductOrServiceModel, a new type may not be needed for them.

What, then, is the bare minimum? At the core of COLD is the idea to describe products through their features ("Specifications" in COLD parlance), using a small set of general properties to make precise the kind of the relationship between the product and a given feature: is it possible, implied, etc.

The idea to describe products through their features is shared by several Schema.org extension proposals[10], and by a planned extension to GR, "ProductFeature"[11], meant to allow the publishing of arbitrary property-value pairs. Basically, they suggest the creation of a new "Feature" type ("Feature", being probably more appropriate than "Specification"), and of a "feature" property.

The exact semantics of this property needs to be precisely defined, however. It should not be used when a feature is only possible; indeed, two features may be possible on a given product while their conjunction is not; also, selecting a possible feature may increase the price. Therefore, using this property for features that are only possible results in an ambiguity regarding the description of PDPs. On the other hand, cold:chosenSpec and cold:impliedSpec can safely be made subProperties of it. Getting back to our earlier example, we may write:

[10] http://thematix.com/ontologies/travel/lodging/Feature.html

[11] http://wiki.goodrelations-vocabulary.org/Documentation/Product_features

```
foo:Model1DieselConf :feature foo:Model1, foo:Diesel, foo:Manual.
foo:Model1GasConf :feature foo:Model1, foo:Gasoline;
    cold:possible [a cold:ConfigurationLink;
        cold:specToBeAdded foo:Manual;
        cold:linkedConf foo:Model1GasManualConf];...
```

Consumer applications, knowing only the Schema.org extension, can now obtain the structured info they need to correctly index these configurations by their features, while those knowledgeable about COLD can grab more; namely, the links to the refined configurations and the precise semantics attached to those links; in other words, they can crawl the refined configurations in a smart way.

The “cold:possible” property and the “ConfigurationLink” class may be too complex to be accepted by Schema.org. Workarounds are possible: search engines can always follow the corresponding hypertext links in the HTML instead, and producers of data can always point search engines to the configurations they regard as interesting, using the sitemap file of their websites.

However, recent proposals[12,13] suggest Schema.org might be ready to tackle reified properties. We hope it will: that would provide the required framework to represent ConfigurationLinks, and we would then come close to a seamless integration of COLD’s main functionalities.

7 Conclusion

Data about customizable products can be published effectively as Linked Data. Most, if not all, configurator applications on the web could be modified to publish data that way. It gets us accurate descriptions of complex ranges of products, which can be crawled by simple agents. The published data are huge, but they can be effectively indexed by search engines. With minimal additions to Schema.org, the description of configurable products such as cars could gain momentum, opening new opportunities.

References

1. M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management*. Springer-Verlag, 2008.
2. E. Chevalier and F.-P. Servant. Product customization as linked data. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*, ESWC’12. Springer-Verlag, 2012.
3. F. Badra, F.-P. Servant and A. Passant. A Semantic Web Representation of a Product Range Specification based on Constraint Satisfaction Problem in the Automotive Industry. OSEMA Workshop ESWC (2011) http://ceur-ws.org/Vol-748/paper4.pdf

[12] https://www.w3.org/wiki/WebSchemas/RolesPattern
[13] https://www.w3.org/wiki/WebSchemas/PropertyValuePairs