ONTOLOGY-DRIVEN PRODUCT CONFIGURATION
Industrial Use Case

Alexander Smirnov, Nikolay Shilov, Alexey Kashevnik
SPIIRAS, St.Petersburg, Russia

Thomas Jung, Mario Sinko, Andreas Oroszi
Festo AG & Co. KG, Esslingen, Germany

Keywords: Product configuration, Ontology, SOA, Industrial environment, Context.

Abstract: The paper proposes an approach based on application of ontology management technology to the tasks of product configuration and product code design in an industrial company. The development of the approach includes usage of Web-services for industrial environment representation and context-based information processing to facilitate the product configuration task. The research is illustrated via a case study for an industrial company that has more than 300 000 customers in 176 countries. A use case for ontology-driven product configuration demonstrates the applicability of the approach.

1 INTRODUCTION

New information technologies open new boundaries for researchers. The service-oriented architecture (SOA) is one such step towards information-driven collaboration. This term today is closely related to other terms such as ubiquitous computing, pervasive computing, smart space and similar, which significantly overlap each other (Balandin et al. 2009). In this paper a conceptual model of ontology-driven product configuration in SOA-based industrial environment is proposed. The SOA is used in order to provide for interoperability.

Current trends in the worldwide economy require companies to implement new production and marketing paradigms. This determines major trends of knowledge-dominated economy: (i) shift from “capital-intensive business environment” to “intelligence-intensive business environment” – an “e” mindset – and (ii) shift from “product push” strategies to a “consumer pull” management – mass customisation approach (Smirnov et al., 2002). A strategy bringing companies and their customers in a closer collaboration is called *innovation democratisation*. This is a relatively new term standing for involvement of customers into the process of designing new products and services. This enables companies to better meet the needs of their customers (von Hippel, 2006).

For companies with wide assortments of products (more than 30 000 – 40 000 products of approx. 700 types, with various configuration possibilities), it is very important to ensure that customers can easily navigate among them. One possible solution is to provide a codification system that can produce easily recognizable and at the same time relatively short codes. This is an important task for customer communication management because well defined and understandable product identification is mandatory for ensuring a good corporate look for the company (Baumeister, 2002; Fjermestad and Romano, 2002; Piller and Schaller, 2002).

The paper proposes an ontology-based approach to product configuration and product code design. Ontologies have shown their usability for this type of tasks (e.g., Bradfield et al., 2007; Chan and Yu, 2007;Patil et al., 2005). A system based on the proposed approach has been implemented in an industrial company that has more than 300 000 customers in 176 countries supported by more than 52 companies worldwide with more than 250 branch offices and authorised agencies in further 36 countries.

Smirnov A., Shilov N., Kashevnik A., Jung T., Sinko M. and Oroszi A..
ONTOLOGY-DRIVEN PRODUCT CONFIGURATION - Industrial Use Case.
DOI: 10.5220/0003635100380047
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2011), pages 38-47
ISBN: 978-989-8425-81-2

2 USE-CASE SCENARIOS

A flexible codification system can be used for different purposes (Figure 1). The first important feature is support for creating new codes for new products. Given the product family and its characteristics an engineer should be able to define the code of this product. Festo sells a wide variety of pneumatic drives. Based on known product type a customer should also be able to define the code of the desirable product based on its characteristics (customer requirements).

The following two basic use cases can be outlined.

2.1 Modular Product Definition

Modular products refer to products, whose functional, spatial and other characteristics fall within a range of possible values. For such products their characteristics are not known in advance. What is known are constraints for the characteristics (e.g., a length can be from 50 to 500 mm with step of 1 mm). It should be possible for a customer to define a valid code for such a product even if it never existed before.

From the customers perspective it is less important if a product is produced from a modular definition or a stock part (except possibility for different delivery times). The important question for customers is to get the product which matches best their requirements.

If a company uses different codification systems for modular and discrete products, its possibility to deliver stock parts instead of modular produced products are limited. The customer usually expects to get the same product, with the same product code, he / she selected during the ordering process.

2.2 Ontology-Driven Product Configuration

On the basis of the formal description of possible products within the common ontology it is possible now to design new applications which offer customers better ways to find and choose the right product.

A simple but always necessary kind of relationship between properties and values describes the consistency of a complex product. This is mainly done by constraints restricting the set of all possible combinations to those which are possible in real-life. The reasons for applying constraints can be different – the most common is the technical possibility of a certain combination.

Furthermore it is possible to add dependant technical data to a certain configuration (which is a set of selected properties and values). For example, a product's weight can be calculated based on the properties / values selected by customer. Another common use case is to configure a CAD 3D model by sending its constructive relevant information from the order code. Practically a lot of data can be made dependant on the current configuration of a modular product. This provides a possibility to provide data which is similarly exact to data of discrete products (for example with a fixed weight).

Even more challenging are inter-product-relationships. The most common use case is the relationship between a main product and an accessory product (Figure 2). While both products are derived from a different complex modular product model there are dependencies which assign a correct accessory to a configured main product. Those dependencies are related to the products individual properties and values. The depth of product-accessory relationships is basically not limited, so accessory-of-accessory combinations have to be taken into account, too.

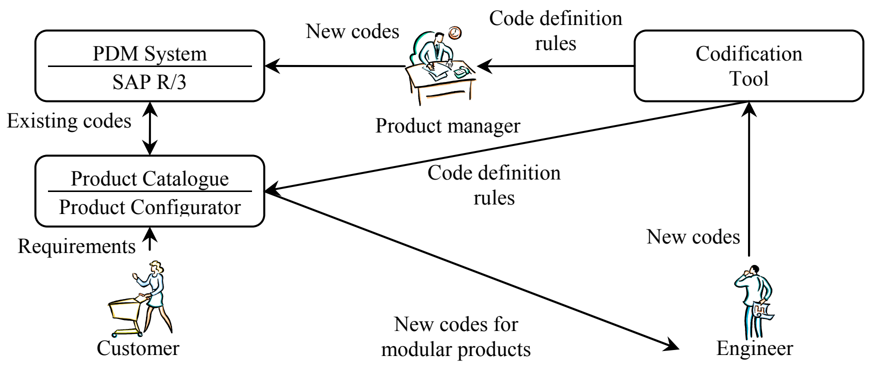


Figure 1: Codification system information flows.

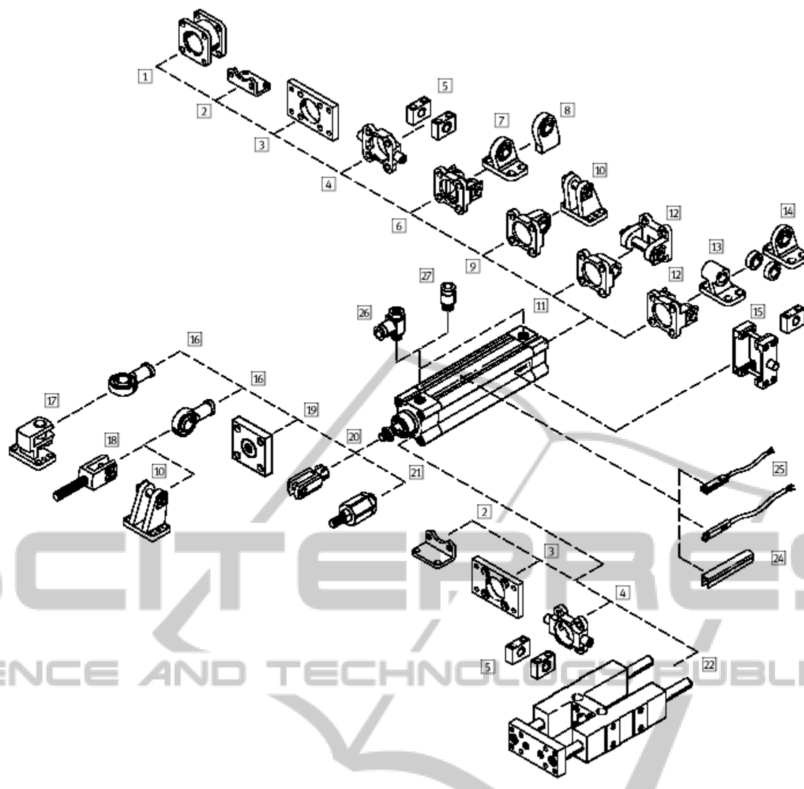


Figure 2: Example for product-accessory-relationship (standard cylinder DNCB, ISO 15552, with assorted accessories).

Certain problems have to be eliminated like circular relationships which lead back to main product. The relationships can be very complex when it comes to define the actual location / orientation of interfaces and mounting points between products.

A more complex scenario is solution-oriented. The idea is to solve a certain real-life problem with modular-products and their inter-product-relationships. The result of such a solution is basically a system of products working together. Used formalism of Object-Oriented Constraint Networks (OOCN, see sec. 3) makes it possible to perform automatic definition of configurable complex products based on the required functions and other constraints specified by the customer.

A handling module offers a good example for this problem. For this purpose the ontology is extended with an extra attribute “Function” for classes.

For illustrative purposes a simple example consisting of three simple products is considered:

- function: movement (Figure 3a)
- function: rotation (Figure 3b)
- function: gripping (Figure 3c)

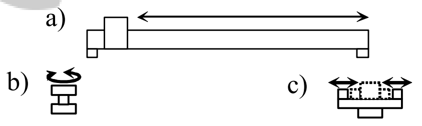


Figure 3: Simple products: (a) movement function, (b) rotation function, (c) gripping function.

The compatibility table for these products is presented in Table 1. The goal of the example is to configure complex products that can perform predefined functions. For instance, if two functions (movement and gripping) are required, then the resulting product will consist of two simple products (Figure 4). If all three functions are required, the resulting product will be as shown in Figure 5. Such a configuration can be calculated automatically by a constraint solver. Principles of the configuration of more complex constructions are planned to be researched.

Table 1: Compatibility table.

	Product 1	Product 2	Product 3
Product 1	-	+	+
Product 2	-	-	+
Product 3	-	-	-

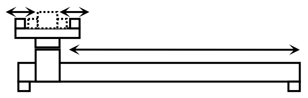


Figure 4: Complex product performing two functions (movement and gripping).

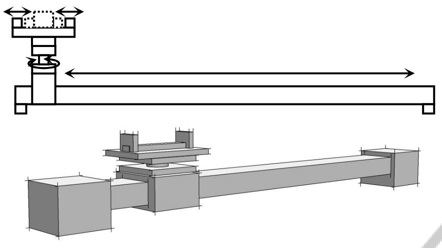


Figure 5: Complex product performing three functions (movement, rotation and gripping).

3 ONTOLOGY-BASED DOMAIN DESCRIPTION

The developed approach is based on the idea that knowledge can be represented by two levels. The first level describes the structure of knowledge (TBox in description logic). Knowledge represented by the second level is an instantiation of the first level knowledge; this knowledge holds object instances (ABox in description logic).

The knowledge of the first level (structural knowledge) is described by a common ontology of the company's product families (classes). Ontologies provide a common way of knowledge representation for its further processing. They are considered as content theories about the sorts of objects, properties of objects and relations between objects that are possible in a specified knowledge domain. They give potential terms for describing the knowledge about the domain (Chandrasekaran et al., Gruber, 1993; Guarino, 1997). Ontology is useful in creating unique models of problem domains by developing specialized knowledge bases specific for various configuration problem domains. It can be defined as an explicit specification of the structure of a certain domain. It includes a vocabulary for referring to the subject area, and a set of logical statements expressing the constraints existing in the domain and restricting the interpretation of the vocabulary.

In this particular case the entities are product families. Usage of product families enables the definition of product platforms that can be reused across whole families of similar products.

In the approach the ontological model is described using the formalism of Object-Oriented Constraint Networks (OOCN). Application of constraint networks allows simplification of the formulation and interpretation of real-world problems which in the areas of management, engineering, manufacturing, etc. are usually presented as constraint satisfaction problems (e.g., Baumgaertel, 2000). This formalism supports declarative representation, efficiency of dynamic constraint solving, as well as problem modelling capability, maintainability, reusability, and extensibility of the object-oriented technology. In the presented approach the product information is supposed to be interpreted as a dynamic constraint satisfaction problem (CSP).

Compatibility of CSP, ontology, and OOCN models is achieved through identification of correspondences between the primitives of these models. The CSP model consists of three parts: (i) a set of variables; (ii) a set of possible values for each variable (its domain); and (iii) a set of constraints restricting the values that the variables can simultaneously take. Typical ontology modelling primitives are classes, relations, functions, and axioms. The formalism of OOCN describes knowledge by sets of classes (product families), class attributes (properties of the products), attribute domains (possible values for the properties), and constraints (explained below). The concept "class" in OOCN notation is introduced instead of concept "object" in the way object-oriented languages suggest. The set of constraints consists of constraints describing "class, attribute, domain" relations; constraints representing structural relations as hierarchical relationships "part-of" and "is-a", classes compatibility, associative relationships, attribute cardinality restrictions; and constraints describing functional dependencies.

The OOCN paradigm defines the common ontology notation used in the system. According to this representation an ontology (A) is defined as: $A = (O, Q, D, C)$ where: O – a set of *object classes* ("*classes*"); each of the entities in a class is considered as an *instance* of the class. Q – a set of class attributes ("*attributes*"). D – a set of attribute domains ("*domains*"). C – a set of *constraints* (Figure 6).

For the chosen notation the following six types of constraints have been defined $C = C^{I} \cup C^{II} \cup C^{III} \cup C^{IV} \cup C^{V} \cup C^{VI}$: $C^{I} = \{c^{I}\}$, $c^{I} = (o, q)$, $o \in O$, $q \in Q$ – accessory of attributes to classes; $C^{II} = \{c^{II}\}$, $c^{II} = (o, q, d)$, $o \in O$, $q \in Q$, $d \in D$ – accessory of domains to attributes; $C^{III} = \{c^{III}\}$, $c^{III} = (\{o\}, True \vee False)$, $|\{o\}| \geq 2$, $o \in O$ – classes compatibility (compatibility structural constraints);

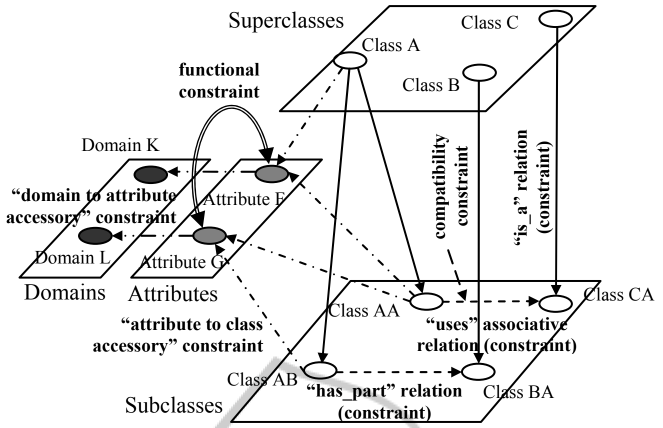

Superclasses
Class C
Class A
Class B
functional constraint
Domain K
"domain to attribute accessory" constraint
Attribute F
Domain L
Attribute G
compatibility constraint
"is_a" relation (constraint)
Domains
Attributes
Class AA
Class CA
"attribute to class accessory" constraint
"uses" associative relation (constraint)
Class AB
"has_part" relation (constraint)
Class BA
Subclasses

Figure 6: Object-oriented constraint network paradigm.

$C^{IV} = \{c^{IV}\}$, $c^{IV} = \langle o', o'', type\rangle$, $o'\in O$, $o''\in O$, $o' \neq o''$ – hierarchical relationships (hierarchical structural constraints) "is a" defining class taxonomy (*type*=0), and "has part"/"part of" defining class hierarchy (*type*=1);$C^V = \{c^V\}$, $c^V = (\{o\})$, $|\{o\}| \geq 2$, $o\in O$ – associative relationships ("one-level" structural constraints); $C^{VI} = \{c^{VI}\}$, $c^{VI} = f(\{o\}, \{o, q\}) = True \vee False$, $|\{o\}| \geq 0$, $|\{q\}| \geq 0$, $o\in O$, $q\in Q$ – functional constraints referring to the names of classes and attributes.

Below, some example constraints are given:

the attribute *Locking in end positions* (q_1) belongs to the class *Series C (pneumatic drive)* (o_1): $c^I_1 = (o_1, q_1)$;

the attribute *Locking in end positions* (q_1) belonging to the class *Series C* (o_1) may take the values *Without (Standard)*, *Extend / Retract*, *Extend*, and *Retract* (the explanation of the values is given in sec. 4): $c^{II}_1 = (o_1, q_1, \{$*Without (Standard)*; *Extend / Retract*; *Extend*; and *Retract*$\})$;

the class *Valve* (o_2) is compatible with the class *Series C* (o_1): $c^{III}_1 = (\{o_1, o_2\}, True)$;

an instance of the class *Valve* (o_2) can be a part of an instance of the class *Valve terminal* (o_3): $c^{IV}_1 = \langle o_2, o_3, 1\rangle$;

the Series C (o_3) is a Pneumatic Drive (o_4): $c^{IV}_1 = \langle o_3, o_4, 0\rangle$;

an instance of the class *Valve* (o_2) can be connected to an instance of the class *Series C* (o_1): $c^V_1 = (o_2, o_1)$;

the value of the attribute *cost* (q_2) of an instance of the class *solution* (o_5) depends on the values of the attribute *cost* (q_2) of instances of the class *component* (o_6) connected to that instance of the class *solution* and on the number of such instances: $c^{VI}_1 = f(\{o_6\}, \{(o_5, q_2), (o_6, q_2)\})$.

4 ONTOLOGICAL MODEL IMPLEMENTATION

The first step to an implementation of the approach is creation of the ontology described above. This operation was done semi-automatically based on existing electronic documents and defined rules of the model building. The resulting ontology consists of more than 1000 classes organized into a 4 level taxonomy, which is based on the VDMA (Verband Deutscher Maschinen- und Anlagenbau - German Engineering Federation) classification (Figure 7). Taxonomical relationships support inheritance that makes it possible to define more common attributes for higher level classes and inherit them for lower level subclasses. The same taxonomy is used in the company's PDM (Product Data Management) and ERP (Enterprise Resource Planning) systems.

For each product family (class) a set of properties (attributes) is defined, and for each property its possible values and their codes are defined as well. The lexicon of properties is ontology-wide, and as a result the values can be reused for different families. Application of the common single ontology provides for the consistency of the product codes and makes it possible to instantly reflect incorporated changes in the codes.

The ontology and the lexicon of properties are multilingual. Each class and attribute can be assigned several names in different languages so that specialists from different countries could work simultaneously while still preserving consistency of the ontology. Additionally, metadata fields and comments can be defined to provide for an additional description of the classes, properties and property values (Figure 8).

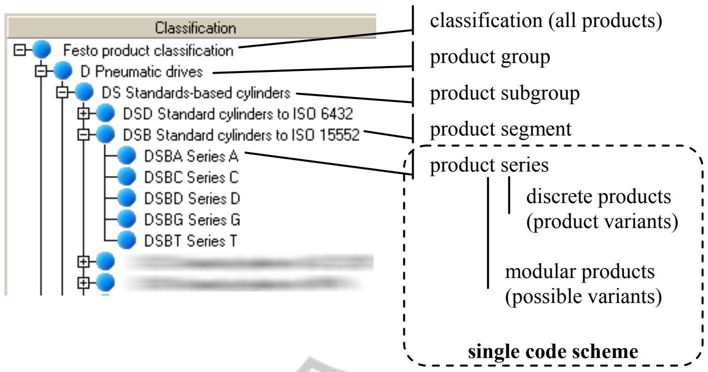
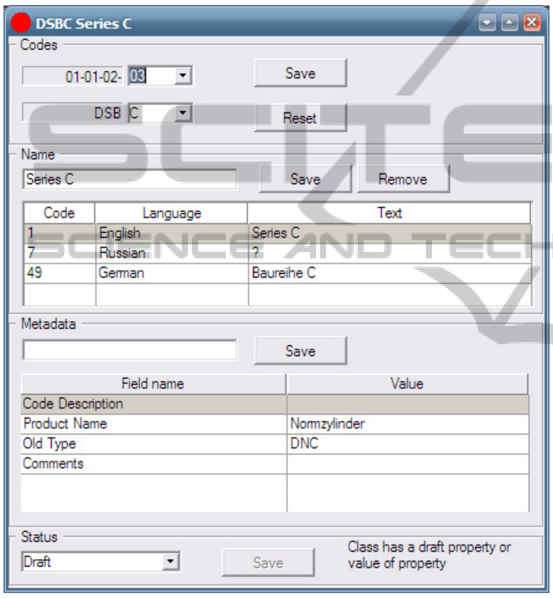

Classification
Festo product classification
D Pneumatic drives
DS Standards-based cylinders
DSD Standard cylinders to ISO 6432
DSB Standard cylinders to ISO 15552
DSBA Series A
DSBC Series C
DSBD Series D
DSBG Series G
DSBT Series T
classification (all products)
product group
product subgroup
product segment
product series
discrete products (product variants)
modular products (possible variants)
single code scheme

Figure 7: Ontology - Festo Product Classification.


DSBC Series C

Codes

01-01-02- 03 Save

DSB C Reset

Name

Series C Save Remove

Code	Language	Text
1	English	Series C
7	Russian	?
49	German	Baureihe C

Metadata

Save

Field name	Value
Code Description	
Product Name	Normzylinder
Old Type	DNC
Comments	

Status

Draft Save Class has a draft property or value of property

Figure 8: Class description example.

5 CODIFICATION RULES DEFINITION

The ontology described above provides rules for the codification system in the following way. For each class a number of attributes is assigned in a certain sequence. This sequence of attributes forms a template for codes of products belonging to the appropriate product family. For each product the properties are replaced with codes of their values corresponding to the particular product to generate its code.

To illustrate this idea the following example can be considered.

The DSBC series is a family of pneumatic drives with a single moving rod (Figure 9). There are 35 different properties each with between 4 to 10 values. Customer and engineers can select from these properties and values. Some combinations of different properties and values are not allowed because they are technically impossible.

Figure 9: DSBC product: a pneumatic drive.

Typical properties of the DSBC series are “locking in end positions” and “special antifriction features”. Their possible values / codes are presented in Table 2.

Given code template consisting of a delimiter and the above two properties, the following codes can be built:

- Standard (no locking, no antifriction features): DSBC
- “Extend / retract” (locking in both directions) and standard antifriction features: DSBC-E1
- “Extend / retract” and reduced friction: DSBC-E1L

Figure 10 represents a fragment of the real code scheme for the DSBC series.

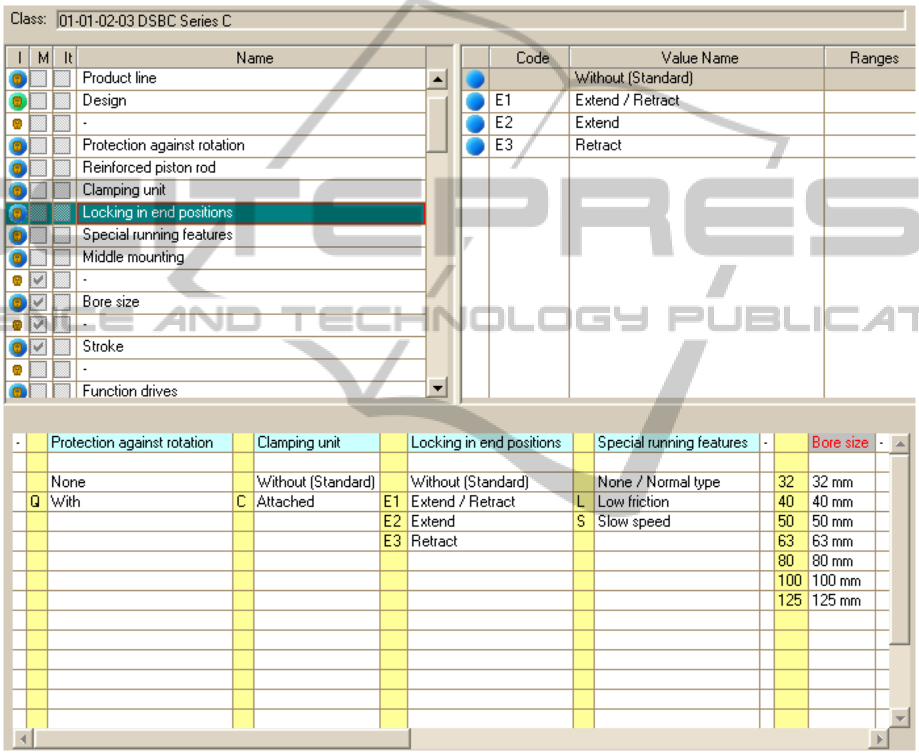
Inheritance of more common properties from higher, more abstract classes ensures that for different branches of the classification the sequence of common properties will be the same. This simplifies the code interpretation.

For example, the “locking in end positions” property is inherited from the DS class (2nd level, product subgroup in the classification) and it can be used in the same relative position in child classes of the DS class.

To avoid ambiguous code interpretations a special validation algorithm has been developed.

Table 2: Compatibility table.

Code	Value Name	Description
Locking in end positions		
	Without (Standard)	the most common choice, no locking in end positions
E1	Extend / Retract	locking is applied in both directions of movement
E2	Extend	locking is applied in the direction of extension
E3	Retract	locking is applied in the direction of retraction
Special antifriction features		
	None / Normal type	the most common choice, no special antifriction features
L	Low friction	product with reduced friction
S	Slow speed	product oriented to slow movement



Class: 01-01-02-03 DSBC Series C

I	M	It	Name
			Product line
			Design
			-
			Protection against rotation
			Reinforced piston rod
			Clamping unit
			Locking in end positions
			Special running features
			Middle mounting
	✓		-
	✓		Bore size
	✓		-
	✓		Stroke
			-
			Function drives

	Code	Value Name	Ranges
●		Without (Standard)	
●	E1	Extend / Retract	
●	E2	Extend	
●	E3	Retract	

-		Protection against rotation		Clamping unit		Locking in end positions		Special running features	-		Bore size	-
		None		Without (Standard)		Without (Standard)		None / Normal type		32	32 mm	
	Q	With	C	Attached	E1	Extend / Retract	L	Low friction		40	40 mm	
					E2	Extend	S	Slow speed		50	50 mm	
					E3	Retract				63	63 mm	
										80	80 mm	
										100	100 mm	
										125	125 mm	

Figure 10: Code scheme for the DSBC series (a fragment).

Built codes and values name can be used for different purposes. Some examples are:

- Order Code Schemas in printed catalogues (selections of properties -> order code)
- Online configurator applications (selections of properties -> order code)
- Order code interpretation / validation (order code -> list of selected properties -> order fulfillment)

6 CONCLUSIONS

Generating a new order code for a new series with the help of the developed system tool takes approximately one day. The technical options presented by the product manager and developer are converted into order-relevant options. As most of the characteristics can be used again, only new options must be discussed and entered in the system. Without the system this process would need several days. Besides this, the error risk would be very large. It could happen that for the same option another code letter is used for example.

The major advantages of the developed system are:

- Systematic order codes for all products;
- Machine readability;
- Quick orientation;
- Security when selecting and ordering products.

One other advantage is the reusability of the data. The structured data are used in other processes such as:

- Automatically creating master data in SAP models;
- Automatically creating data for the configuration models;
- Automatically generating an ordering sheet for the print documentation (this ordering sheet was generated earlier with high expenditure manually);
- Automatically generating a product list which is needed in the complete process implementing new series.

7 FUTURE WORK: COLLABORATIVE SOA-BASED INDUSTRIAL ENVIRONMENT

The approach is based on the idea of characterizing all members and components of the industrial environment by their roles (e.g., designer, production manager, production facility, etc.) and of describing them via profiles. These profiles are associated with agent-based services that negotiate in order to take into account explicit and tacit preferences of the industrial environment components.

This approach integrates efficient sharing of information with a service-oriented architecture taking into account the dynamic nature of the industrial environment. For this purpose the models proposed are actualized in accordance with the current situation. An ontological model is used in the approach to solve the problem of service heterogeneity. This model makes it possible to enable interoperability between heterogeneous information services due to provision of their common semantics and terminology (Uschold and Grüninger, 1996). Application of the context model makes it possible to reduce the amount of information to be processed. This model enables management of information relevant for the current situation (Dey, 2001) (e.g., the current industrial segment). The access to the services, information acquisition, transfer, and processing (including integration) are performed via usage of the technology of Web-services.

This work is work-in-progress and the paper proposes conceptual ideas for their further development. Figure 11 represents the generic scheme of the SOA-based industrial environment representation.

The main idea of the approach is to represent the members and components of the industrial environment by sets of services provided by them. This makes it possible to replace the information sharing between them with that between distributed services. For the purpose of interoperability the

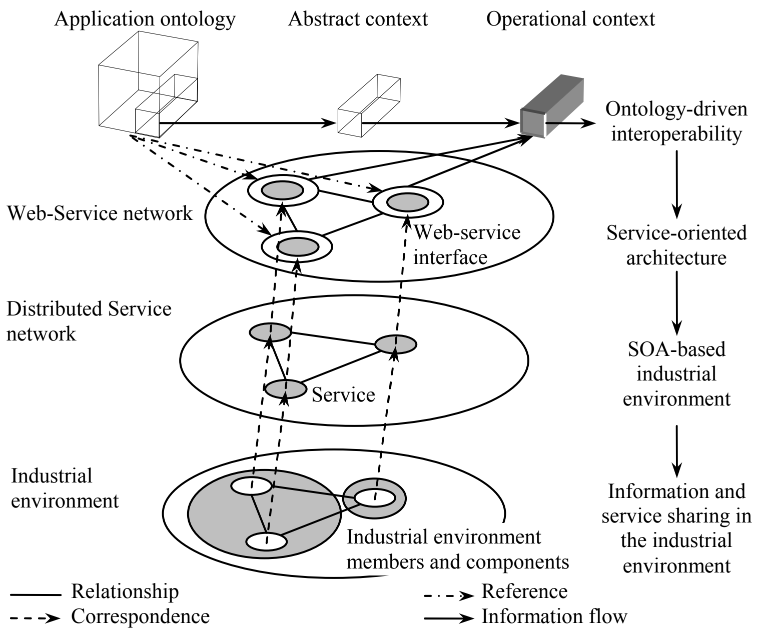

Application ontology
Abstract context
Operational context
Ontology-driven interoperability
Web-Service network
Web-service interface
Service-oriented architecture
Distributed Service network
Service
SOA-based industrial environment
Industrial environment
Industrial environment members and components
Information and service sharing in the industrial environment
Relationship
Correspondence
Reference
Information flow

Figure 11: Generic scheme of the approach.

services are represented by Web-services using the common notation described by the application ontology. Depending on the considered problem the relevant part of the application ontology is selected forming the abstract context that, in turn, is filled with values from the sources resulting in the operational context.

The service-oriented architecture has a number of advantages resulting from its principles (CADRC, 2009). Among these the following should be mentioned (the specifics related to the industrial environment are indicated in italics):

1 **Service Autonomy.** Services engineered for autonomy exercise a high degree of control over their underlying run-time execution environment. Autonomy, in this context, represents the level of independence which a service can exert over its functional logic. *With regard to the industrial environment the autonomy also reflects independence of the network members, which in real life are often have different subordination.*

2 **Service Abstraction.** Further supporting service autonomy, SOA advocates that the scope and content of a service's interface be both explicitly described and limited to that which is absolutely necessary for the service to be effectively employed. Beyond the service interface, abstraction applies to any information, in any form, describing aspects of the service's design, implementation, employed technologies, etc. *This principle helps to abstract from real services provided by the industrial environment members and components and concentrates on their modelling via Web-services.*

3 **Service Standardisation.** As services are typically distributed throughout networks, they must be easily accessible by other entities in terms of discoverability and consequential invocation. Given this requirement, service-oriented architecture recommends that services adhere to standards, including, for example, standards for the language used to describe a service to prospective consumers. *In the proposed approach the standardisation is achieved via usage of the common standards such as WSDL and SOAP as well as common terminology described by the application ontology. As a result the services constituting the network are fully interoperable and can communicate with each other without any problems.*

4 **Service Reusability.** Reusability is a central property of any successful service. It denotes the capacity of a service to be employed in support of not just one but rather a variety of business models. SOA promotes such functional reuse through stipulations for service autonomy and interface abstraction. With these features, the same service can be invoked by multiple consumers, operating in various business domains, without requiring the service provider to re-code service internals for each application domain. *Service reusability significantly facilitates the modelling process and decreases the amount of work required for model building. Besides, the existing services of the industrial network members and components can be used.*

ACKNOWLEDGEMENTS

The presented work is a result of the joint project between SPIIRAS and Festo Ag&Co KG. Some parts of the research were carried out under projects funded by grants # 09-07-00436, # 10-07-00368 and # 09-07-00066 of the Russian Foundation for Basic Research, and project #213 of the research program "Intelligent information technologies, mathematical modelling, system analysis and automation" of the Russian Academy of Sciences.

REFERENCES

Balandin, S., Moltchanov, D., Koucheryavy, Y., eds., 2009. *Smart Spaces and Next Generation Wired/Wireless Networking*, Springer, LNCS 5764.

Baumeister, H., 2002. Customer relationship management for SMEs, *Proceedings of the 2nd Annual Conference eBusiness and eWork e2002*, Prague, Czech Republic.

Baumgaertel, H., 2000. Distributed constraint processing for production logistics, IEEE Intelligent Systems, 15(1) 40-48.

Bradfield, D. J., Gao, J. X., Soltan, H., 2007. A Metaknowledge Approach to Facilitate Knowledge Sharing in the Global Product Development Process, *Computer-Aided Design & Applications*, 4(1-4) 519-528.

CADRC, 2009. KML Sandbox: An Experimenation Facility Based on SOA Principles. *CADRD Currents*, Fall, 2009, Collaborative Agent Design Research Center (CADRC), California Polytechnic State University, San Luis Obisro.

Chan, E. C. K., Yu, K. M., 2007. A framework of ontology-enabled product knowledge management, *International Journal of Product Development*, Inderscience Publishers, 4(3-4) 241-254.

Chandrasekaran, B., Josephson, J. R., Benjamins, V. R., 1999. What are Ontologies, and Why Do We Need Them?,*IEEE Intelligent Systems & Their Applications*, (January/February), 20-26.

Dey, A. K., 2001. Understanding and Using Context, *Personal and Ubiquitous Computing J.*, 5(1) 4-7.

Fjermestad, J., Romano, N. C., Jr., 2003. An Integrative Implementation Framework for Electronic Customer Relationship Management: Revisiting the General Principles of Usability and Resistance, *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, Big Island, HI, USA.

Gruber, T. R. 1993. A translation approach to portable ontologies, *Knowledge Acquisition*, 5(2) 199-220.

Guarino N. 1997. Understanding, Building, and Using Ontologies – A Commentary to "Using Explicit Ontologies in KBS Development" (by van Heijst, Schreiber, and Wielinga), *International Journal of Human and Computer Studies*, 46(2/3) 293-310.

von Hippel, E., 2006. *Democratizing innovation*, The MIT Press, Cambridge, Massachussets, USA.

Piller, F., Schaller, C., 2004. Individualization based Collaborative Customer Relationship Management: Motives, Structures, and Modes of Collaboration for Mass Customization and CRM, *Working Paper No. 29 of the Dept. of General and Industrial Management,* Technische Universität München.

Patil, L., Dutta, D., Sriram, R., 2005. Ontology-based exchange of product data semantics, *IEEE Transactions on Automation Science and Engineering*, 2(3) 213-225.

A. Smirnov, A., Pashkin, M., Chilov, N., Levashova, T., 2002. Knowledge Fusion in the Business Information Environment for e-Manufacturing Pursuing Mass Customisation, *Moving into Mass Customization. Information Systems and Management Principles* (eds. C. Rautenstrauch, R. Seelmann-Eggebert, K. Turowski), Springer, 153-175.

Uschold, M., Grüninger, M., 1996. Ontologies: Principles, methods and applications, *Knowledge Engineering Review*, 11(2) 93-155.